Controlo de operações industriais emuladas em ambientes de realidade híbrida

António Fernandes Postiga

Dissertação de Mestrado

Orientador: Prof. António Pessoa de Magalhães

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Mecânica

Ramo de Automação

Julho de 2018

Resumo

A constante evolução, no sentido positivo, do número de alunos que frequentam a especialização de Automação no curso de Engenharia Mecânica na Faculdade de Engenharia da Universidade do Porto, aliada a uma constante escassez de recursos financeiros, materiais e anos, tem como consequência a impossibilidade de dar aos estudantes formação prática em diversas áreas chave da Automação, sendo por isso algumas delas apenas abordadas quase exclusivamente em termos teóricos.

Tentando colmatar este défice de vivência e aprendizagem experimental, este projeto foi realizado com o intuito de familiarizar o seu autor com alguns equipamentos e protocolos comuns no mundo industrial, permitindo-lhe assim reconhecer não só as possibilidades, mas também os requisitos e limitações destes. Concretamente, a presente dissertação descreve a realização de sistemas de controlo de alguma complexidade para ambientes industriais, envolvendo o recurso a PLC's, HMI's, protocolos de comunicação, motores elétricos, *drives* e *software* de simulação.

As soluções de controlo cobriram primeiramente problemas de ordenação e sequenciamento e foram desenvolvidas sobre ambientes industriais simples e totalmente "sintéticos" (i.e., simulados em computador em tempo real). Todavia, envolveram algoritmos de tomada de decisão relativamente complexos e aplicáveis a numerosos casos práticos. Posteriormente, abordaram-se ambientes e problemas industriais envolvendo o controlo distribuído de movimento, uma questão sempre atual, mas muito mais delicada do ponto de vista da simulação em tempo real, daí o recurso, neste caso, a uma plataforma de teste híbrida, onde os sistemas virtuais interagiram em tempo real com recursos físicos externos ao *software* de simulação.

Os ambientes industriais utilizados foram construídos e simulados no *software* Factory I/O, criado pela empresa Real Games, e os algoritmos de controlo foram criados no *software* Codesys, da empresa 3S-Smart Software Solutions GMBH, e implementado no *soft-PLC* Codesys Control Win V3, da mesma empresa. Os equipamentos físicos controlados resumem-se a dois motores de indução trifásicos cujo controlo foi realizado através da utilização de dois variadores de frequência em comunicação MODBUS RTU com o soft-PLC, e a dois motores passo-a-passo comandados por um PLC da gama S7-1200, da Siemens, em comunicação MODBUS TCP com o soft-PLC.

O resultado mais significativo deste desenvolvimento foi a familiarização do autor com a realidade do controlo de movimento imposto através de PLC's e com a implementação de código de controlo de acordo com a norma IEC 61131-3. Além disto, a FEUP obteve um sistema que pode, eventualmente, ajudar futuros alunos a compreender melhor como funciona a linguagem de texto estruturado e como a utilizar de acordo com o GRAFCET, ferramenta que aprenderam a usar para modelar sistemas de eventos discretos.

Foram dadas algumas sugestões à Real Games no que toca a correção de *bugs*, diversificação de alguns objetos e implementação de funcionalidades extra.

Palavras-chave: Automação industrial; sistema educacional; simulação fabril; PLC; HMI; controlo de movimento; rede industrial.

Abstract

The constant evolution, in the positive sense, of the number of students attending the Automation specialization during the Mechanical Engineering course at the Faculty of Engineering of the University of Porto, together with a constant shortage of financial, material and human resources, results in the impossibility to give the students practical training in several key areas of Automation, many of them being approached only in theorical terms.

To fill this deficit of experience and experimental learning, this project was designed to familiarize its author with some the most common equipment and protocols in the industrial world, allowing him to recognize not only the possibilities, but also the requirements and limitations of these. Specifically, this dissertation describes the development of control systems of some complexity for industrial environments, involving the use of PLC's, HMI's, communication protocols, electric motors, drives and simulation software.

The control solutions first covered sorting problems and were developed over simple and fully "synthetic" industrial environments (i.e., real-time computer simulated). However, they involved the use of relatively complex decision-making algorithms that were applicable to many practical cases. Subsequently, industrial environments and problems regarding distributed motion control, a subject that is never outdated, were dealt with. Given this subject's delicacy from a real-time simulation point of view, a hybrid test platform was used, where the virtual systems interacted in real time with physical, real resources, external to the simulation.

The environments used were built and simulated in the software Factory I/O, created by the company Real Games, and the control algorithms were created in the software Codesys, owned by the company 3S-Smart Software Solutions GMBH, and implemented in the soft-PLC Codesys Control Win V3, owned by the same company. The controlled physical equipment consists of two three-phase induction motors, controlled by a frequency inverter each, that communicated in MODBUS RTU with the soft-PLC, and two stepper motors controlled by a PLC, belonging to the S7-1200 product range by Siemens, in MODBUS TCP with the soft-PLC.

The most significant result of this development was the author's familiarization with the reality of motion control imposed by PLC's and the implementation of control code according to the IEC 61131-3 standard. In addition to this, the faculty has obtained a system that may eventually help future students to better understand how structured text language works and how to use it in agreement with GRAFCET, a tool they have learned the model discrete event systems.

Some suggestions have been to Real Games regarding bug fixes, diversification of some objects and implementation of extra features.

Keywords: Industrial automation; educational system; factory simulation; PLC; HMI; motion control; industrial network.

Agradecimentos

Ofereço os meus honestos agradecimentos ao Professor António Pessoa de Magalhães, pela oportunidade que me deu para desenvolver os meus conhecimentos e capacidades em diversos temas que considero de elevada importância e interesse. Agradeço-lhe também pela motivação e compreensão que demonstrou ao longo deste trabalho.

Agradeço ao senhor Bruno Teófilo Vigário e a toda a equipa da Real Games, por se terem mostrado disponíveis para ajudar alguém que nada tinha para dar em troca, sempre de uma forma muito profissional. Os seus esforços foram cruciais para o desenvolvimento do projeto tal como tinha sido idealizado.

Expresso também a minha mais sincera gratidão ao senhor Ramalho pelo ótimo trabalho que realizou.

Por fim, agradeço também à minha família e amigos, que me apoiaram sempre com vontade não só ao longo deste projeto, mas também de todo o curso.

Índice

Índice de figuras

Índice de tabelas

Lista de siglas e acrónimos

- EN – *European Standard*
- FEUP – Faculdade de Engenharia da Universidade do Porto
- GRAFCET – *Graphe Fonctionnel de Commande des Étapes et Transitions*
- HMI – *Human Machine Interface*
- I/O – *Input/Output*
- IDE – *Integrated Development Environment*
- IEC – *International Electrotechnical Commission*
- OLE – *Object Linking and Embedding*
- OPC – *OLE for Process Control*
- OSCAT – *Open Source Community for Automation Technology*
- PLC – *Programmable Logic Controller*
- RS (232/422/485) – *Recommended Standard*
- RTU – *Remote Terminal Unit*
- TCP – *Transmission Control Protocol*
- TIA – *Totally Integrated Automation*

1. Introdução

1.1. Motivação e objetivos

O presente trabalho trata a conceção e implementação, em autómatos programáveis, de sistemas de controlo para ambientes industriais, e foi realizado no âmbito de uma dissertação de mestrado em engenharia mecânica, na Faculdade de Engenharia da Universidade do Porto.

A motivação para a realização de uma dissertação sobre o tema enunciado teve como origem o interesse e necessidade sentidos pelo autor em reunir experiência sobre *software* e equipamentos utilizados em ambiente industrial, e sobre a forma como estes comunicam entre si, de forma a terminar o curso com uma base mais sólida que lhe permitisse entrar no mercado de trabalho de forma mais suave.

Essa ambição do autor correspondia precisamente ao objetivo da tese tal como proposta pelo seu supervisor: dar uma formação mais avançada ao estudante que o aceitasse, complementando a obtida ao longo do curso, no desenvolvimento e implementação de soluções de controlo de ambientes industriais. Nesse quadro encaixaram-se o desenvolvimento de modelos que definam e caraterizem sistemas de eventos discretos, assim como a lógica inerente ao controlo destes, a materialização dessa lógica em programas para PLC's de uma forma coerente e facilmente relacionável com o modelo e o estabelecimento de comunicações com outros equipamentos tanto a nível físico como lógico.

Nesse sentido tornou-se importante, primeiramente, definir e criar cenários industriais de interesse, com aplicação prática relevante e análogos aos observados em instalações reais. Por indisponibilidade de um cenário industrial real, foi utilizado um *software* de simulação de cenários industriais. Em alguns cenários, totalmente sintéticos, não há nenhuma ligação com equipamentos reais, mas noutros optou-se por relacionar certos componentes do cenário virtual com equipamentos reais, formando assim cenários híbridos.

Seguidamente, o segundo objetivo foi o de criar modelos que traduzissem o comportamento de várias partes de cada cenário, e gerar pogramas para PLC's que implementassem o comportamento especificado nesses modelos de forma a que o programa e o respetivo modelo se pudessem facilmente relacionar e a conversão entre um e outro fosse expedita.

Em terceiro lugar, ambicionou-se a utilização de componentes ligados à área do controlo de movimento na indústria, mais precisamente motores elétricos, efetuando-se o seu controlo através de PLC's.

Finalmente, o objetivo final foi o de estabelecer comunicações entre PLC's e equipamentos, reais e virtuais, fazendo uso de protocolos de comunicação conhecidos, de forma a que tudo funcionasse em uníssono de acordo com o comportamento desejado.

1.2. Linhas orientadoras do trabalho a realizar

De maneira a atingir os objetivos propostos pelo autor no capítulo anterior, este decidiu que seria lógico e útil tratar casos exemplo que espelhassem, pelo menos em parte, a realidade do mundo industrial atual. Para tal, optou-se por escolher situações clássicas, transversais a muitos ramos da indústria, com o intuito de conferir ao trabalho realizado um caráter mais genérico. De entre a imensidão de opções existentes, a escolha incidiu em duas: transporte e manipulação de objetos ao longo de instalações industriais e controlo de movimento.

1.2.1. Transporte e manipulação de objetos

O transporte e manipulação de objetos ao longo de uma instalação é um tema que, embora não seja o mais vistoso pela sua complexidade, é sem dúvida global e importante. No sentido de abordar este tema planeou-se a criação e controlo de cenários industriais, nos quais o objetivo do controlo é o transporte e manipulação de objetos segundo determinadas regras.

O transporte, no seu significado mais primitivo, é apenas a deslocação de algo de um ponto inicial para um final. Claro está que, num ambiente industrial, isto tem de ser realizado de forma metódica, e é aí que entra a importância da manipulação associada ao transporte. Pois ao longo do processo de transporte torna-se muitas vezes necessário ajustar a posição, reorientar o objeto, ou mudar a direção do seu movimento para garantir que este chegue ao local pretendido nas devidas condições, sejam elas quais forem.

Numa situação mais simples esta manipulação pode ser feita sempre da mesma maneira, independentemente do objeto em causa. Outras vezes depende de uma ou várias caraterísticas, identificativas do objeto, a partir das quais se toma a decisão de como o manipular. Um exemplo seria um caso de separação no qual cada objeto, chegando a uma bifurcação, seria manipulado de maneira a seguir o devido caminho face às suas caraterísticas. Por vezes a manipulação depende não das caraterísticas um único objeto, mas sim das de um conjunto deles, sendo um perfeito exemplo de uma situação deste género um caso de ordenação.

Para tratar estes tipos de manipulação associada ao transporte foram criados e controlados vários cenários industriais, que preveem a aplicação de lógicas de ordenação (capítulo 4) e de separação (capítulo 5).

1.2.2. Controlo de movimento

O controlo de movimento é outro tema que, sendo clássico, é sempre atual. É um tema muito heterogéneo, na medida em que a maneira como se aplica o controlo de movimento depende muito do sistema a controlar: número de graus de liberdade; número de atuadores; tipo de atuadores; etc… e de como se o quer controlar: anel aberto; anel fechado com controlador PID; anel fechado com SMC (*sliding mode control*); etc…

Como praticamente todas as máquinas realizam algum tipo de movimento, a aplicação do controlo de movimento abrange-se a máquinas elétricas, pneumáticas, hidráulicas, térmicas e outras. Mesmo focando só um destes tipos, por exemplo, as máquinas elétricas, mais precisamente os motores elétricos, é possível identificar uma

grande variedade de motores, com diferentes princípios de funcionamento, tais como: motores DC; motores AC síncronos; motores AC assíncronos; motores passo-a-passo; etc… Isto implica métodos diferentes de alimentação, com consequente utilização de equipamentos dedicados, equipamentos esses que muitas vezes não possuem capacidades de controlo ou comunicação obrigando, portanto, ao uso de outros e adicionando cada vez mais complexidade ao sistema.

Por esta razão pode-se concluir que o tema do controlo de movimento, seja este simples ou complexo, é simplesmente demasiado comum, e como tal não poderia ser ignorado numa dissertação com o objetivo de dar formação ao seu autor. Assim sendo, foram utilizados diferentes tipos de motores elétricos, com diferentes tipos de alimentação e equipamento auxiliar, de forma a abordar este tema (capítulo 5).

1.2.3. Materialização dos sistemas-alvo

Tendo em conta o objetivo de controlar ambientes industriais, o primeiro passo foi decidir como criar tais ambientes. Sendo que a utilização de um ambiente industrial real era irrealista - na medida, por exemplo, em que seria demasiado dispendioso para a faculdade, que a sua utilização levantaria inevitavelmente questões de segurança e a realização de testes para os programas de controlo desenvolvidos seria mais limitada - a solução para o problema passou pela utilização de um *software* de simulação de ambientes industriais. Esta foi uma boa solução para o caso do tópico referente ao transporte e manipulação de objetos pois abriu imensas possibilidades, a custo zero, para a idealização de cenários industriais que correspondessem às necessidades para aplicação de lógicas de ordenação e separação.

Se esta foi uma boa solução para o caso referido, pois para o caso do controlo de movimento é uma solução longe de ser adequada. Em primeiro lugar, porque o *software* de simulação de ambientes industriais disponível não contém os equipamentos desejados, isto é, motores elétricos. Mesmo que tivesse, estes seriam, possivelmente, demasiado simples quando comparados com as versões reais. E em segundo lugar, porque se perderia toda uma complexidade inerente ao funcionamento destes dispositivos, que passa por necessidades de: utilização de equipamento auxiliar, como *drives* ou variadores de frequência; utilização de protocolos de comunicação de acordo com as limitações dos equipamentos; e cablagem, que para ser bem feita obriga à interpretação correta de fichas de dados do fabricante. Pelas razões enunciadas, o controlo de movimento foi exercido sobre motores elétricos reais.

1.3. Estrutura da dissertação

Esta dissertação está organizada em 6 capítulos, servindo o presente para a introduzir, e um anexo.

O capítulo 2 – Recursos tecnológicos de suporte – apresenta todos os recursos necessários à realização desta dissertação, referindo-se as funcionalidades de cada um. Assim sendo aborda-se, por um lado, os que são objeto de controlo e, por outro, os que servem para impor o comportamento desejado, assim como os meios que possibilitaram a ligação entre eles.

O capítulo 3 – Modelação e implementação de controladores lógicos – introduz e define adequadamente a ferramenta de modelação selecionada para modelar os sistemas estudados. Para além disso, apresenta também um método desenvolvido pelo autor para a geração de código para PLC a partir dos modelos criados com a ferramenta em questão.

O capítulo 4 – Estudo de casos de ordenação – é um dos dois capítulos principais da dissertação, e trata o estudo de casos em que se pretende ordenar objetos, culminando no desenvolvimento de dois algoritmos: um para ordenação na horizontal e outro para ordenação na vertical. Esses algoritmos foram, a seguir, utilizados em cenários industriais simples, obrigando à implementação dos algoritmos em código para PLC.

O capítulo 5 – Estudo de casos de controlo de movimento – é o segundo dos capítulos principais da dissertação, tratando-se neste o controlo de velocidade de dois tipos de motores elétricos. Esse controlo é efetuado no contexto de um cenário híbrido, onde há uma parte virtual onde se simula uma situação de fabrico e separação de peças, e uma parte real que corresponde ao controlo dos motores elétricos. Além disso, também se trata neste capítulo a criação de uma HMI para alteração de parâmetros do sistema.

O capítulo 6 – Conclusões e trabalhos futuros – encerra a dissertação. Nele se resumem as principais conclusões, se destacam os ensinamentos mais relevantes e se propõem temas de interesse no seguimento do trabalho realizado.

O anexo A – Grafcets técnicos para o controlo do cenário de ordenação horizontal – compreende os grafcets criados a partir do código correspondente a cada programa de PLC desenvolvido para o controlo do cenário de ordenação horizontal.

2. Recursos tecnológicos de suporte

Neste capítulo são referidos os recursos tecnológicos utilizados ao longo desta dissertação. Estes são:

- Simulador de ambientes industriais;
- Controladores lógicos programáveis e *software* de programação;
- Motores;
- Variadores de frequência;
- *Drives*;
- Equipamento auxiliar.

Cada um destes recursos será devidamente abordado a seguir.

2.1. Factory I/O

O programa de simulação de ambientes industriais utilizado foi o Factory I/O. Este *software*, criado e atualmente mantido pela empresa portuguesa Real Games, permite a criação e simulação dos mais variados ambientes industriais recorrendo a um motor de física. Põe ao dispor do utilizador diversos dispositivos típicos de uma instalação industrial, como tapetes e sensores, assim como diversos *drivers* de comunicação para possibilitar a comunicação com os mais diversos controladores.

Para a realização de alguns dos objetivos pretendidos nesta dissertação foi necessário solicitar uma versão especial à Real Games. A empresa facilitou também o acesso à versão 2.3.0 do programa, não lançada até ao momento de redação deste documento, que implementava funcionalidades potencialmente interessantes. Apesar de isto ter ocorrido numa fase mais avançada do projeto, na qual já não havia recursos para fazer uso destas funcionalidades, esta foi a versão utilizada para gerar os cenários virtuais.

2.1.1. Princípio de funcionamento

O programa apresenta dois modos: o modo de edição onde se cria o cenário e se muda os parâmetros dos dispositivos, e modo de funcionamento onde os dispositivos entram em funcionamento segundo as ordens de um controlador e as peças móveis se comportam segundo a leis da física.

Criar o cenário virtual é extremamente simples, basta arrastar um dispositivo da janela de seleção de dispositivos e colocá-lo no lugar desejado do cenário. Depois de colocado pode-se configurar o dispositivo, se este tiver opções de configuração, e o cenário está criado. Este procedimento está exemplificado na Figura 1.

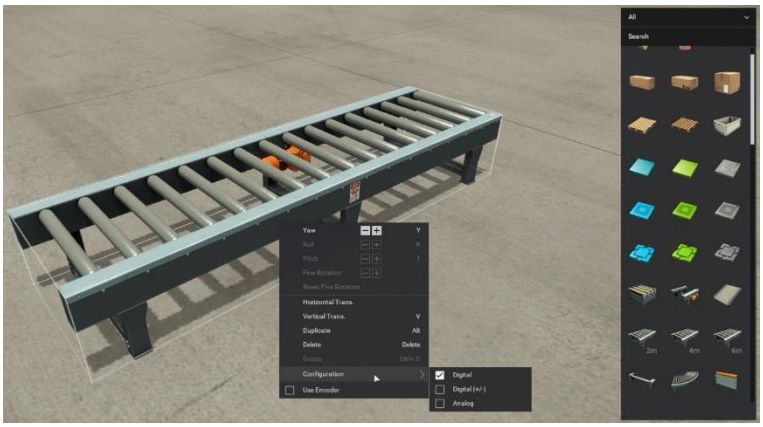

Figura 1 - Introdução de um dispositivo no cenário virtual.

Depois de criado o cenário, e com o controlador pronto, o próximo passo é estabelecer a comunicação entre o Factory I/O e o controlador (um *soft-PLC*, por exemplo). Na janela de *drivers* é possível escolher um de vários disponíveis, e a ligação entre cada atuador e a respetiva variável no controlador é feita de forma intuitiva arrastando o nome do atuador para o nome da variável do controlador como se pode observar no exemplo da Figura 2.

Figura 2 - Configuração de ligações entre o Factory I/O e um servidor OPC.

Colocando o cenário virtual em modo de funcionamento, os valores dos sensores do Factory I/O serão transmitidos ao controlador, definindo o valor das variáveis associadas, e os sinais provenientes do controlador conduzem à operação de cada atuador.

2.1.2. Dispositivos

O programa dispõe de uma variedade de dispositivos para serem usados na construção e funcionamento dos cenários virtuais. Nesta secção serão introduzidos apenas os dispositivos utilizados pelo autor ao longo da dissertação.

2.1.2.1. Emissor e eliminador

Os emissores e os eliminadores são itens com uma função muito simples, mas essencial para os cenários virtuais em que o objetivo é “transportar itens de A para B”. O emissor permite a introdução de itens transportáveis no cenário, e o eliminador permite a remoção desses itens do cenário. Ambos estão ilustrados na Figura 3.

Figura 3 - Emissor e eliminador (da esquerda para a direita) num tapete de rolos.

2.1.2.2. Itens transportáveis

Sendo que em cenários industriais reais há frequentemente produtos que são alvo de transporte ao longo da instalação, o Factory I/O disponibiliza uma multitude de produtos móveis que variam em: dimensões, forma, massa, cor e material de fabrico.

De todos os disponibilizados, foram usados os seguintes:

- Caixa para paletização;
- Material em bruto azul;
- Material em bruto verde;
- Tampa azul;
- Tampa verde;
- Base azul;
- Base verde.

O material em bruto de uma cor pode ser transformado, num centro de maquinagem (do qual se fala um pouco mais à frente), numa tampa ou numa base da mesma cor, dependendo da configuração do centro de maquinagem. Assim, os materiais em bruto, as tampas e as bases foram utilizados num cenário de transformação de produto seguida de separação por forma e cor. Estes itens podem ser observados na Figura 4.

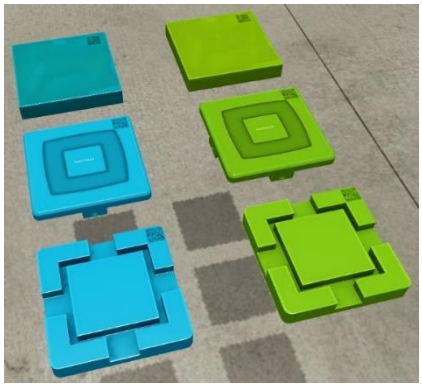

Figura 4 - Material em bruto, tampa e base de cada cor.

As caixas para paletização foram utilizadas em cenários de reorganização, em que o critério de discriminação foi a massa. A massa não era, no entanto, uma opção viável para efetuar a distinção entre as caixas no programa, visto que todas as caixas de paletização tinham uma massa fixa de 3 kg. Para tal feito foi necessária a colaboração da

equipa da Real Games, que se disponibilizou de forma totalmente gratuita para ajudar, e em tempo útil devolveu uma versão ligeiramente diferente do Factory I/O em que as caixas de paletização tinham massas diferentes, que variavam de 2 kg a 18 kg, tendo também cores diferentes de modo a que a diferença de massa pudesse ser feita por um observador. Uma pilha com algumas caixas de cor diferente é mostrada na Figura 5.

Figura 5 - Pilha de caixas para paletização com massas e cores diferentes.

O código de cores para as massas das caixas é o seguinte:

- Cinzento – 18 kg;
- Violeta – 16 kg;
- Azul – 14 kg;
- Verde – 12 kg;
- Amarelo – 10 kg;
- Laranja – 8 kg;
- Vermelho – 6 kg;
- Castanho – 4 kg;
- Preto – 2 kg.

Na versão 2.3.0 existem peças semelhantes às azuis e verdes, feitas em metal.

2.1.2.3. Tapetes transportadores

Os tapetes transportadores são dispositivos muito comuns em instalações industriais e foram largamente utilizados no projeto desenvolvido. São atuadores simples que têm como propósito deslocar itens numa direção. Existem em três tamanhos, variando apenas o comprimento entre eles, sendo que este pode tomar o valor de 2 m, 4 m ou 6 m. Um exemplar de cada um pode ser visto na Figura 6.

Figura 6 - Tapetes de tela de 2 m, 4 m e 6 m.

No seu estado mais simples, configuração digital unidirecional, o tapete apenas se pode mover num único sentido a velocidade constante ou estar parado, obedecendo a uma variável booleana. No modo digital bidirecional o funcionamento é semelhante, sendo que neste é possível mover o tapete nos dois sentidos. Existe ainda um modo analógico, no qual é possível controlar a velocidade e sentido do tapete, que são definidos em resposta a uma variável real compreendida no intervalo [-10; +10].

Na versão 2.3.0 os tapetes possuem um sinal de saída opcional que simula um *encoder*, dando informação sobre a posição do tapete.

2.1.2.4. Separador por rodízios

Um separador por rodízios (*pop up wheel sorter*) serve para mudar a direção de deslocamento de um item, permitindo enviar esse item do um tapete para outro que esteja orientado de outra forma. Os rodízios podem ser orientados 45º para a direita, 45º para a esquerda ou manterem-se na posição de 0º. Quando o separador está desativado, os rodízios mantêm-se numa posição mais baixa e um item que chegue ao separador desliza por cima deste. Quando o separador está ligado os rodízios sobem e rodam, e um item que lá chegue é empurrado na direção que os rodízios tomam. Na Figura 7 pode observar-se um separador por rodízios (no estado desligado).

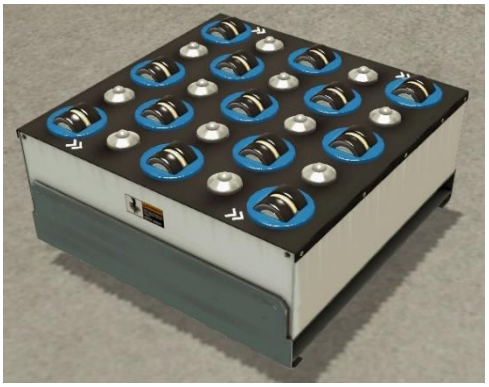

Figura 7 - Separador por rodízios.

Como os rodízios podem girar nos dois sentidos, o separador por rodízios pode ser utilizado não só para separar itens, mas também para convergi-los e juntá-los num único tapete.

2.1.2.5. Manipulador cartesiano

O manipulador cartesiano (*pick and place*), apesar de não ser muito vulgar na indústria, é de extrema utilidade nos cenários virtuais do Factory I/O pois permite fazer uma manipulação vertical, algo que não possível fazer com quase nenhum dispositivo existente no programa. A Figura 8 apresenta um destes manipuladores.

Figura 8 - Manipulador cartesiano.

O manipulador move os seus braços na horizontal ou vertical entre as posições extremas se estiver configurado no modo digital, ou para qualquer posição se estiver no modo analógico, no qual responde a uma variável real de valor compreendido no intervalo [0; 10]. O manipulador segura o item com uma animação de sucção, que só pode ser ligada ou desligada. É possível obter sinais analógicos da posição dos braços e um sinal digital referente à deteção de um item agarrado.

Na versão 2.3.0 o manipulador ganha dois graus de liberdade, podendo girar sob o pilar de apoio e podendo girar a ventosa que segura o item.

2.1.2.6. Pushers

Os *pushers* (Figura 9) são atuadores muito simples cuja função é empurrar itens. Estes dispositivos têm várias configurações possíveis, sendo estas:

- Monoestável;
- Monoestável rápido;
- Biestável;
- Analógico.

As primeiras três opções correspondem a configurações digitais, respondendo o dispositivo a variáveis booleanas. As configurações monoestável e monoestável rápido diferem entre si apenas na velocidade do atuador, sendo que ambas avançam o recuam o atuador em resposta a uma variável booleana. Na configuração biestável o dispositivo requer duas variáveis booleanas, uma que dá a ordem para avançar e outra a ordem para recuar. A velocidade de avanço e recuo é semelhante à da configuração monoestável (não rápida). Com a configuração analógica é possível escolher a velocidade de avanço e de recuo com uma variável real de valor contido no intervalo [-10; +10].

Nas configurações digitais o dispositivo devolve duas variáveis booleanas, uma para cada extremo, que se tornam verdadeiras quando o atuador chega a esse extremo. Na configuração analógica o dispositivo devolve apenas uma variável analógica, de valor contido no intervalo [0; 10], correspondente à posição do atuador.

Figura 9 - Pusher.

2.1.2.7. Posicionadores

Os posicionadores são dispositivos pensados para serem utilizados em tapetes, e cuja função é orientar e posicionar itens no tapete. A sequência mais vulgar é a seguinte:

- Acolher um item;
- Fechar o posicionador para alinhar o item;
- Abrir e levantar o posicionador para deixar o item passar.

Há dois tipos de posicionadores, muito semelhantes: os de posicionamento à direita e os de posicionamento à esquerda. Na Figura 10 pode-se ver um posicionador de posicionamento à direita no estado aberto e levantado, e um posicionador de posicionamento à esquerda no estado fechado e baixado.

Figura 10 - Posicionador à direita levantado e posicionador à esquerda fechado.

Estes dispositivos são de comando exclusivamente digital, sendo possível fechar e abrir o posicionador com uma variável booleana e levantá-lo com uma outra variável do mesmo tipo. Devolvem duas variáveis booleanas: uma que é verdadeira quando o posicionador levanta totalmente e uma que é verdadeira quando o posicionador fecha totalmente ou faz força contra um item.

2.1.2.8. Lâmina de paragem

A lâmina de paragem é um dispositivo, normalmente instalado entre tapetes, que impede a passagem de itens mesmo com o tapete a funcionar. Normalmente serve para acumular e compactar itens num tapete. Este dispositivo é extremamente simples, e a única variável de que dispõe é uma variável booleana de entrada para ativação. Na Figura 11 é mostrado um destes dispositivos.

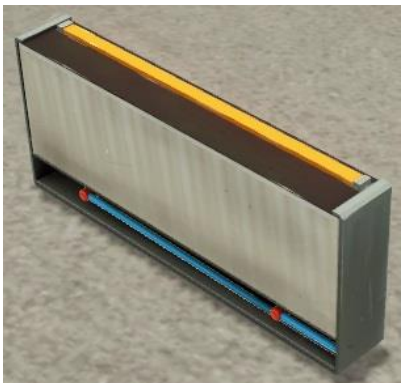

Figura 11 - Lâmina de paragem.

2.1.2.9. Centro de maquinagem

O centro de maquinagem é um equipamento especial na Factory I/O, na medida em que tem a capacidade de transformar e manipular automaticamente peças. O objetivo primário do centro de maquinagem é transformar os itens "Material em bruto" de uma cor em produtos finalizados que são sempre "Tampas" ou "Bases" da mesma cor. O centro de maquinagem, com todos os seus constituintes, pode ser observado na Figura 12.

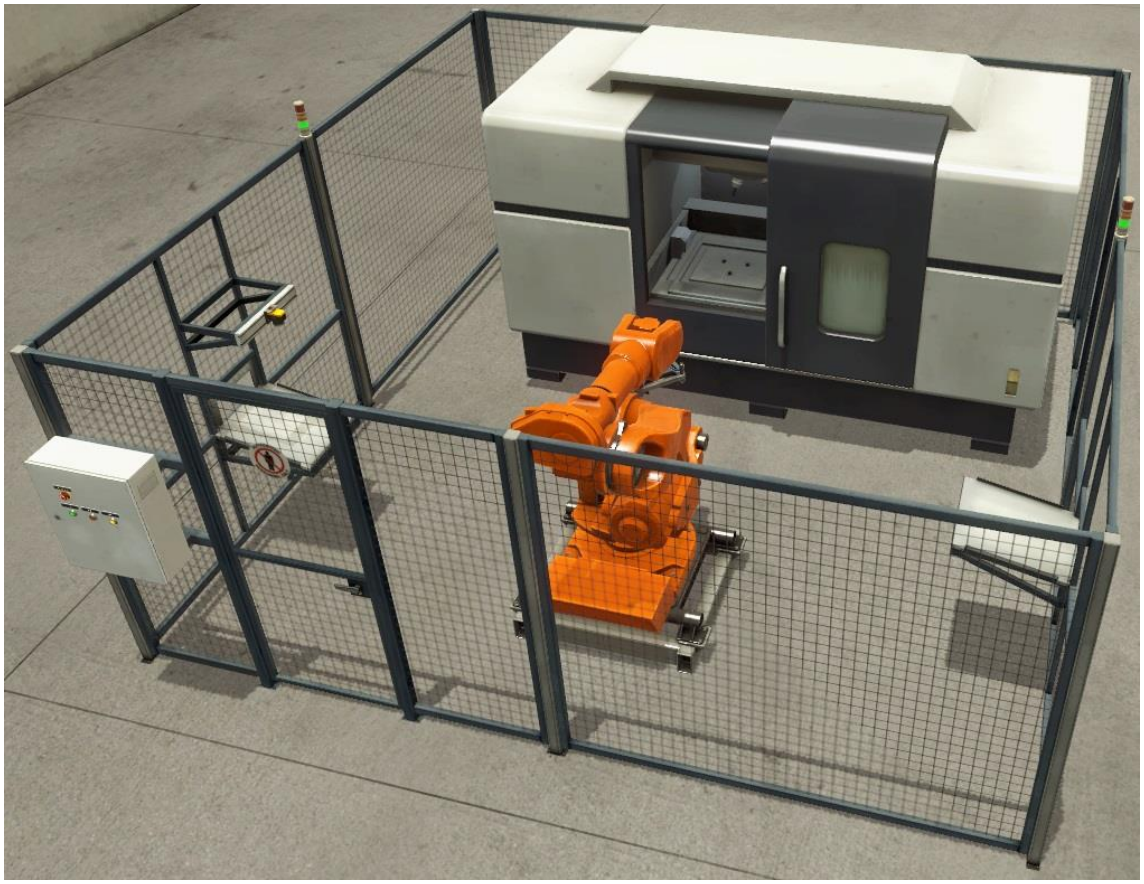

Figura 12 - Centro de maquinagem.

Alimentando o centro de maquinagem com uma peça válida, ou seja, um material em bruto, o robot irá agarrar automaticamente na peça e colocá-la dentro de uma máquina CNC. A porta da máquina CNC fecha e volta a abrir após um tempo predeterminado, que depende do tipo de peça a fabricar, e o *robot* volta a agarra na peça, agora acabada, e

coloca-a na saída do centro de maquinagem. Colocar um item que não seja material em bruto na entrada do centro de maquinagem causará um erro. O tempo de produção das tampas é de 6 s e o das bases é de 3 s.

O centro de maquinagem não tem opções configuráveis sendo, portanto, as entradas e saídas variáveis sempre do mesmo tipo, como indicado a seguir:

Entradas

- Start (bool)
- Stop (bool)
- Reset (bool)
- Produzir tampas (bool)

Saídas

- Ocupado (bool)
- Erro (bool)
- Aberto (bool)
- Progresso (real)

As variáveis de entrada "Start", "Stop" e "Reset" são autoexplicativas. Normalmente o centro de maquinagem é ligado uma vez no início da simulação e não é mais desligado. A variável "Produzir tampas" serve para indicar ao centro de maquinagem, quando uma peça entra na máquina CNC, se essa peça deve ser transformada numa tampa ou numa base.

A variável de saída "Ocupado" toma o valor verdadeiro quando existe uma peça em processamento dentro do centro de maquinagem, estando essa peça a ser transformada ou simplesmente a ser manipulada. A variável "Erro" indica um erro do centro de maquinagem, possivelmente por este ter sido alimentado com uma peça indevida. A variável "Aberto" índica se a porta da máquina CNC está aberta ou fechada, ou seja, acaba por indicar se uma peça está a ser transformada, assim como a variável "Progresso", que toma um valor entre 0 e 100 que traduz o progresso da transformação da peça que está dentro da máquina CNC.

Por necessidade de controlar o tempo de transformação das peças dentro da máquina CNC, foi mais uma vez pedido ajuda à equipa da Real Games. A solução encontrada consistiu em criar uma variável booleana, que pode ou não ser ativada nas configurações, e que termina a transformação da peça quando sofre uma transição ascendente. Implícito está que neste modo de funcionamento a variável "Progresso" deixa de fazer sentido, pelo simples facto de que o programa não sabe quando o controlador vai decidir que a transformação foi concluída.

2.1.2.10. Sensores

O Factory I/O possui uma variedade de sensores que inclui seis tipos, dos quais foram utilizados apenas três (Figura 13):

- Sensor difuso;
- Barreira de luz;
- Sensor de visão.

O sensor difuso é o mais simples e foi o mais usado. O seu funcionamento é muito simples: o sensor devolve uma variável booleana com valor verdadeiro se encontrar um objeto dentro do alcance definido, caso contrário devolve essa variável com o valor falso. Tem um alcance máximo de 1,6 m, o que foi mais suficiente para as aplicações em questão.

A barreira de luz pode ser encarada como uma série de sensores difusos, oito para ser exato. Este dispositivo também tem várias configurações possíveis no que toca à maneira como devolve a informação. No modo digital devolve oito variáveis booleanas, cada uma correspondente a cada um dos seus sensores, tal como se fossem oito sensores difusos colocados independentemente. No modo numérico devolve uma *word*, em que cada *bit* corresponde a um dos sensores. E no modo analógico devolve um valor real diretamente proporcional ao número de sensores que estão a detetar algo.

O sensor de visão é sem dúvida o mais versátil pois permite não só detetar um item, mas também identificá-lo. No modo de configuração digital o sensor devolve três variáveis booleanas, que em conjunto definem o tipo de item detetado. No modo numérico o sensor devolve um valor inteiro que identifica o tipo de objeto detetado. Há ainda o modo de identificação por ID em que o sensor devolve um valor inteiro, único para o objeto em causa, que foi definido quando o objeto apareceu em cena.

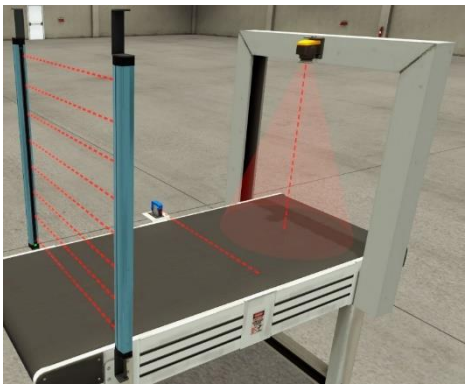

Figura 13 - Barreira de luz, sensor difuso e sensor de visão (da esquerda para a direita).

2.1.3. Drivers

O Factory I/O possuiu uma multitude de *drivers* que permitem a comunicação com os mais diversos PLC's, seja por comunicação direta com estes, por um servidor, através de uma placa de aquisição de dados ou fazendo uso de protocolos de comunicação abertos.

Ao longo desta dissertação, o único *driver* utilizado foi o que permite a ligação do Factory I/O a um servidor OPC e, portanto, este será o único a ser abordado.

2.1.3.1. OPC

Depois de configurado o servidor OPC, algo que não é feito neste programa, efetuar a ligação é relativamente fácil. Na janela de configuração basta procurar servidores disponíveis, carregando em "Browse servers", e após a pesquisa estar concluída escolher o servidor desejado. Em seguida é preciso sondar o servidor em busca de variáveis, e para tal basta carregar no botão "Browse items", concluindo assim a configuração.

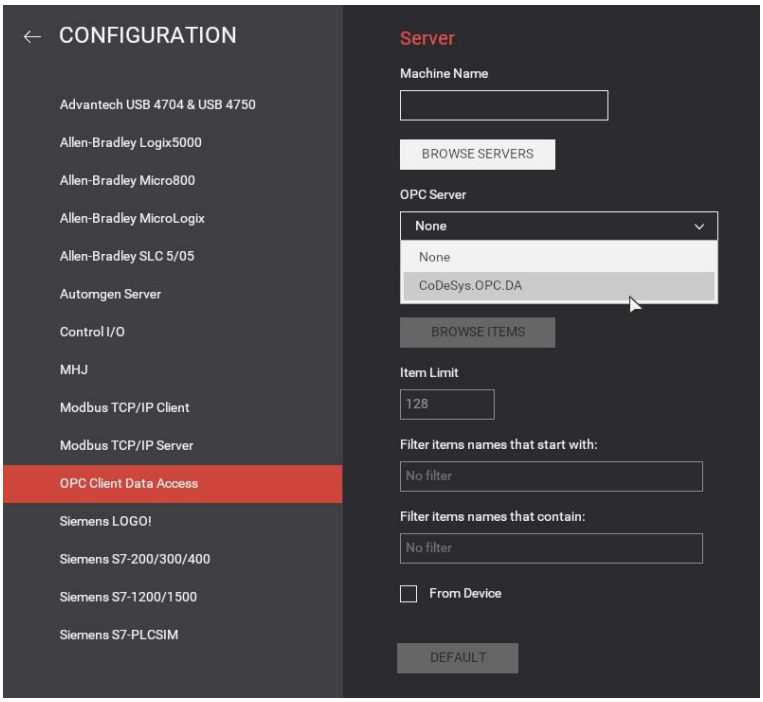

Figura 14 - Configuração do servidor OPC no Factory I/O.

2.2. Controladores lógicos programáveis e *software* de programação

O uso de controladores lógicos programáveis era essencial na realização desta dissertação, tendo em conta os objetivos propostos. Os PLC's representam uma parte fulcral da automação, e por isso o seu uso era obrigatório no controlo de cenários industriais.

A escolha dos PLC's a utilizar dependeu primariamente sobre os seguintes fatores:

- Disponibilidade financeira;
- Relevância num contexto industrial;
- Atualidade do *software* de programação associado.

Em relação ao primeiro ponto, a solução deveria recair em dispositivos físicos já existentes na faculdade. No entanto, para além desta, existe outra solução, que passa por recorrer a *software* que faz uso dos recursos de um computador para correr programas para PLC's, os chamados *soft-PLC*'s. Entre os primeiros e os segundos estão os PLC's mais relevantes da atualidade.

No que toca ao terceiro ponto, qualquer PLC cujo *software* de programação fosse atual e/ou de uso comum seria uma boa escolha, na medida em que permitiria a familiarização com esses *softwares*, o que à partida seria útil, pelo menos a longo prazo.

2.2.1. Codesys e Control Win V3

O ambiente de desenvolvimento integrado (IDE) Codesys é um *software* gratuito para desenvolvimento de programas para PLC, criado e mantido pela empresa *3S-Smart Software Solutions GmbH*. Este *software* permite, para além do desenvolvimento de programas, a criação de interfaces visuais, vulgarmente conhecidas por HMI's, e inclui

ainda um *soft-PLC* denominado *Codesys Control Win V3* e um servidor de OPC no pacote de instalação. O *soft-PLC*, na sua versão de demonstração gratuita, só pode ser executado por duas horas antes de se desligar automaticamente sendo, contudo, possível voltar a ligá-lo imediatamente.

Neste trabalho foi usada a versão V3.5 SP12 Patch 3 do Codesys, na medida em que esta implementava uma funcionalidade para as HMI's que se revelou particularmente útil. Do *soft-PLC* Control Win V3 usaram-se duas versões: versão 3.5.12.30 x64 e versão 3.5.10.10 [1].

2.2.1.1. Interface

Considera-se que o *software* de programação tem uma interface simples e relativamente intuitiva. Na Figura 15 está apresentado o exemplo de uma árvore que inclui todos os elementos do projeto dispostos de forma hierárquica, tais como: dispositivos; programas; funções; listas de variáveis globais; comunicações configuradas; etc…

Figura 15 - Exemplo de árvore hierárquica no Codesys.

2.2.1.2. Linguagens de programação compatíveis

O *soft-PLC* Control Win V3 é altamente compatível com a norma IEC 61131-3 e por isso o Codesys permite a criação de programas em diversas linguagens previstas na norma, sendo estas:

- *Instruction list;*
- *Sequencial function chart;*
- *Function block diagram;*
- *Ladder logic diagram;*
- *Structured text.*

A mais usada ao longo deste trabalho foi a *structured text* – texto estruturado – por ser, sem sombra de dúvida, a mais versátil, e por obedecer à norma.

Foram utilizadas algumas funções adicionais, escritas maioritariamente em texto estruturado, de uma biblioteca de utilização gratuita criada por terceiros, denominada OSCAT.

2.2.1.3. Criação de interfaces de visualização

A criação de interfaces de visualização também é relativamente simples. Escolhendo um item para adicionar, por exemplo um botão ou um sinalizador luminoso, segue-se a configuração de parâmetros relativos a esse item: posição no ecrã, tamanho, cor, etc. Muito frequentemente associam-se itens no ecrã a variáveis no programa, de forma a que estes itens reflitam o estado da variável associada ou permitam que se mude esse estado. É possível realizar tarefas mais complexas, associadas a eventos (p. ex. clicar num item), tais como abrir uma janela ou correr um determinado código. Um exemplo de uma HMI criada no Codesys, onde se vê as opções para um item selecionado, pode ser visto na Figura 16.

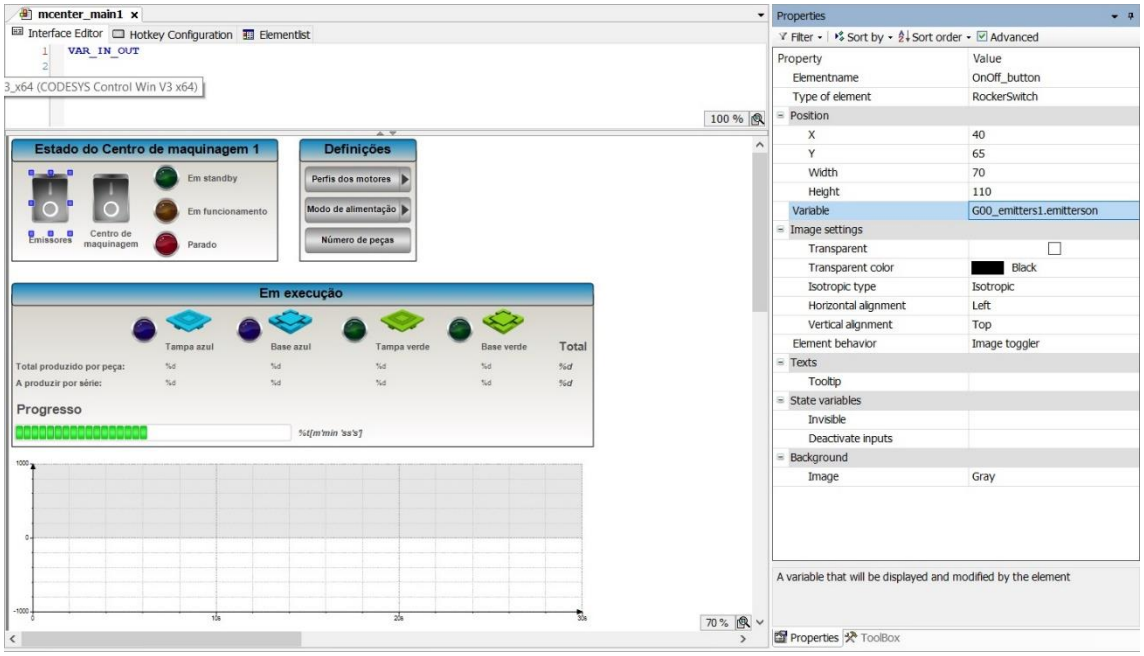

Figura 16 - Criação de interface de visualização no Codesys.

2.2.1.4. Capacidades de comunicação

Em termos de capacidades de comunicação, pode-se dizer que o *soft-PLC* é extremamente versátil, desde que tenha o *hardware* de suporte, sendo capaz de realizar comunicações por vários protocolos, dos quais foram utilizados:

- Modbus RTU – mestre e/ou escravo;
- Modbus TCP – cliente e/ou servidor;
- OPC.

2.2.2. TIA Portal e SIMATIC S7-1200

SIMATIC S7-1200 é uma gama de PLC's programáveis através do *software* STEP 7 - TIA Portal, ambos criados pela Siemens AG. Tal como o Codesys, o TIA Portal também permite a criação de programas para PLC e de HMI's.

Neste trabalho foi usado o *software* STEP 7 – TIA Portal BASIC V13 para programar o PLC, que foi um SIMATIC S7-1200, modelo CPU 1214C AC/DC/Rly, com o módulo gerador de impulsos a alta frequência SB 1222 DC 200 kHz [2].

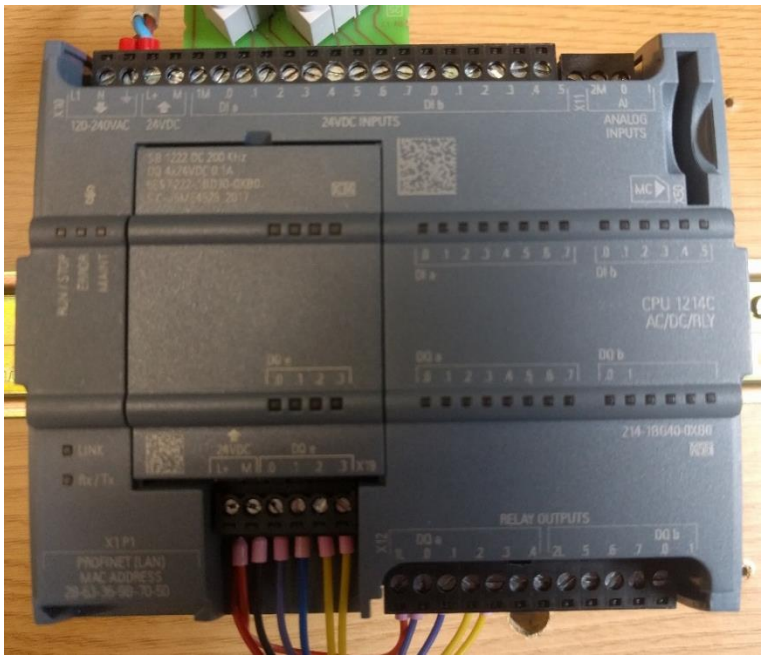

Figura 17 - PLC SIMATIC S7-1200 utilizado.

2.2.2.1. Ambiente de programação

A interface do *software* de programação é bem mais pesada do que a do Codesys. Existe também uma árvore hierárquica com os mais diversos itens (Figura 18), de forma semelhante ao Codesys, mas o *software* no geral é mais difícil e menos intuitivo de usar, para além de que é menos responsivo.

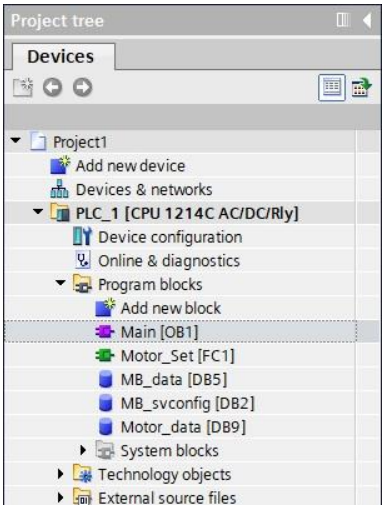

Figura 18 - Exemplo de árvore hierárquica do TIA Portal V13.

2.2.2.2. Linguagens de programação compatíveis

Os PLC's da gama S7-1200 suportam as seguintes linguagens de programação:

- *Function block diagram;*
- *Ladder logic diagram;*
- *Structured control language.*

A linguagem *structured control language* é baseada na norma IEC 61131-3, sendo por isso muito semelhante à linguagem estruturada usada no Codesys. Neste PLC a linguagem mais utilizada no trabalho foi *ladder*.

2.2.2.3. Capacidades de comunicação

Tal como o *soft-PLC* já tratado, o S7-1200 está fortemente dependente de *hardware* extra para poder realizar comunicações que não sejam por Ethernet. São possíveis comunicações por vários protocolos, dos quais o único utilizado foi o MODBUS TCP.

2.3. Motores elétricos utilizados

Dada a necessidade de ter componentes reais para controlar e não só os cenários virtuais, foi escolhida uma série de motores para serem alvo de controlo. De entre os escolhidos tem-se dois motores de indução trifásicos diferentes e dois motores passo-a-passo iguais.

2.3.1. Motores trifásicos

Um dos motores trifásicos utilizados foi um motor de 0,18kW do fabricante ABB. O motor é um M2VA63B-4, e uma foto deste com a respetiva placa de identificação encontra-se na Figura 19.

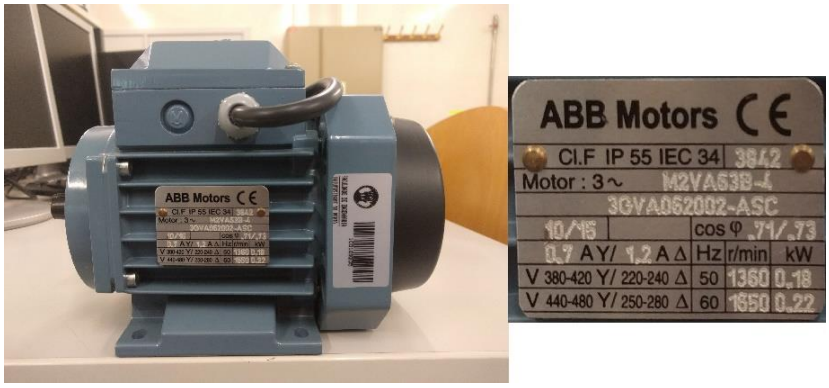

Figura 19 - Motor trifásico de 180W da ABB com placa de identificação.

O outro motor trifásico utilizado foi um motor de 0,37kW do fabricante Sati. O motor é um M71b4, e uma foto deste com a respetiva placa de identificação encontra-se na Figura 20.

Figura 20 - Motor trifásico de 370W da Sati com placa de identificação.

2.3.2. Motores passo-a-passo

Os motores passo-a-passo utilizados são motores de 48 passos por rotação do fabricante eMinebea. São motores PM55L-048, e uma foto destes é apresentada na Figura 21.

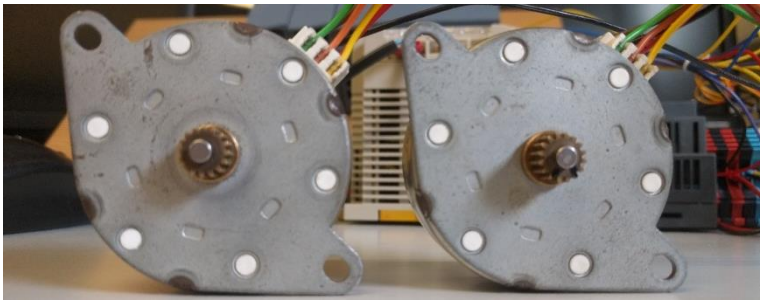

Figura 21 - Motores passo-a-passo da eMinebea.

2.4. Variadores de frequência

De modo a ser possível controlar os motores de indução trifásicos foi necessário a utilização de variadores de frequência, não só para variar a velocidade de rotação do motor, mas também para efetuar a ligação ao PLC. Os variadores escolhidos (Figura 22) são da gama VARISPEED J7, da Omron, sendo que um deles tem uma potência de 0,1kW (J7AZB0P1) e o outro tem uma potência de 0,55kW (J7AZB0P4). Ambos têm a capacidade de alojar um módulo para comunicação em RS-485 ou RS-422, que pode ser utilizado para realizar comunicações em MODBUS RTU.

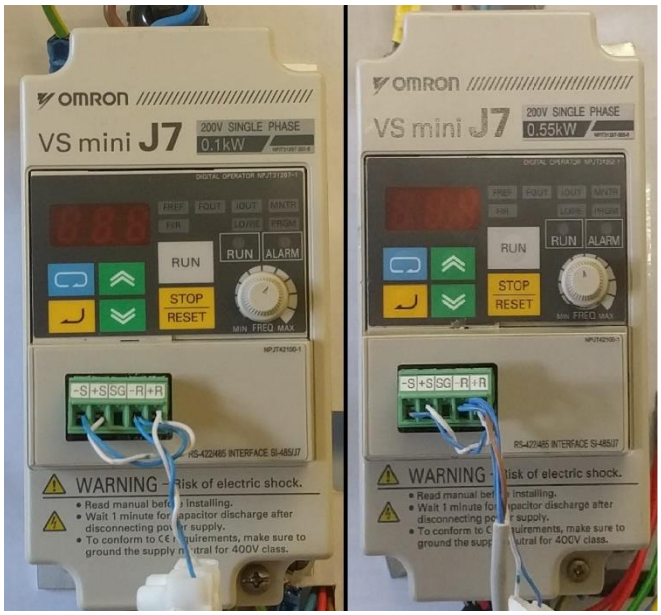

Figura 22 - Variadores de frequência de 0,1kW e 0,55kW (da esquerda para a direita).

2.5. Drives

Por natureza, os motores passo-a-passo precisam de um *drive* para poderem funcionar de maneira correta. Estes *drives* recebem um sinal de impulsos e, por cada impulso, energizam as bobines de forma a que o motor dê um passo. Para além do sinal

de impulsos, o *drive* também recebe um sinal de sentido de rotação, um sinal de ativação do motor e um sinal para alternar entre passo completo e meio passo. Os *drives* escolhidos, um para cada motor, são dois SAMOTRONIC101 (4 636 6608 0) do fabricante Saia-Burgess Controls AG, e podem sem observados na Figura 23.

Figura 23 - Drive para os motores passo-a-passo.

2.6. Equipamento auxiliar

Depois de expostos todos os componentes principais utilizados ao longo da realização deste trabalho resta mencionar algum equipamento de apoio à construção de redes de comunicação que também foi utilizado, sendo esse:

- Adaptador USB-RS232;
- Adaptador RS232-RS485;
- Fonte de alimentação de 24 V;
- Regulador de tensão de 12 V.

3. Modelação e implementação de controladores lógicos

No desenvolvimento de sistemas industriais torna-se importante especificar e documentar, de forma clara e concreta, determinadas caraterísticas importantes para a definição desses sistemas. Para atingir tal objetivo é comum a utilização de modelos. Um modelo é uma representação de algum sistema ou objeto dentro de um determinado meio. Um modelo capta os aspetos importantes daquilo que modela sob um certo ponto de vista, simplificando ou omitindo o acessório [3]. A utilização de modelos como forma de descrever um sistema é útil não só na fase de projeto, contexto mais comum no qual sistemas foram abordados ao longo desta dissertação, mas também numa possível situação futura de diagnóstico ou em que se pretenda modificar, por alguma razão, o sistema original.

Os sistemas industriais abordados nesta dissertação enquadram-se, assim como muitos outros, na definição de "sistema de eventos discretos". Um sistema de eventos discretos pode ser definido como um sistema dinâmico com estados discretos em que as transições entre estes são desencadeadas por eventos [4]. Estes eventos podem ser ocorrências simples e genéricas, como a deteção de uma peça por um sensor, ou o cumprimento de condições complexas ou que resultem de algoritmos complexos. O primeiro caso é de tal forma genérico que se torna quase impossível criar um sistema industrial que não contenha exemplos de tal, mas o segundo nem sempre acontece, e foi no sentido de o expor que alguns dos cenários foram estudados.

Quando se trata de programar controladores para o controlo de sistemas de eventos discretos o modelo do sistema a controlar pode e deve ser usado como mapa para a geração do código para o controlador. A adoção de uma metodologia adequada para este procedimento permite uma programação mais fácil e um código de tal forma coerente com o modelo que é possível recriar o modelo a partir do código.

Essa metodologia varia com a ferramenta utilizada para criar o modelo e com a linguagem de programação utilizada. Nesta dissertação foi escolhido o GRAFCET (Graphe Fonctionnel de Commande des Étapes et Transitions), definido na norma EN 60848 [5], como linguagem de modelação de sistemas de eventos discretos, e o texto estruturado, definido na norma IEC 61131-3 [6, 7], como linguagem de programação do controlador.

3.1. Modelação em GRAFCET

A grande maioria dos sistemas automáticos pode ser encaixada numa das seguintes categorias: combinacionais ou sequenciais. O comportamento dos sistemas combinacionais depende, em cada instante, unicamente das suas variáveis de entrada. Já os sistemas sequenciais dependem não só das suas variáveis de entrada, mas também do seu estado atual.

Este segundo tipo de sistema por ser definido por um simples diagrama de estados, mas estes diagramas não são suficientemente poderosos para descrever sucintamente sistemas mais complexos (categoria na qual se encaixam a maior parte dos sistemas

automáticos), podendo tornarem-se demasiado caóticos e consequentemente difíceis de utilizar. Foi no sentido de colmatar esta dificuldade que foi criado o GRAFCET.

O GRAFCET é uma poderosa linguagem gráfica de modelação, baseado nas redes de Petri (que é por sua vez uma linguagem de modelação também), com origem na França, em 1977, e tinha como objetivo a especificação de comportamentos sequenciais e especialmente controladores lógicos [8].

3.1.1. Estrutura e interpretação

Neste trabalho será apresentado apenas um breve, mas julgado suficiente, resumo do que é o GRAFCET, de forma a que o leitor possa entender os modelos criados. O GRAFCET permite descrever e especificar de forma consistente e inequívoca o comportamento de sistemas sequenciais mais complexos, como os encontrados na indústria. Este objetivo é atingido pelo uso de gráficos, aos quais também se chamam Grafcets, que representam a evolução do sistema através de *etapas* e *transições*, às quais podem ser associadas *ações*. Um exemplo muito básico de um Grafcet com duas etapas, uma transição e uma ação pode ser encontrado na Figura 24. Os seguintes três subcapítulos estão de acordo com [9].

3.1.1.1. Etapas

Uma etapa representa parte do estado de um sistema. Uma etapa pode estar ativa ou inativa, sendo que o conjunto de etapas ativas representa o estado do sistema num determinado instante. Uma etapa é representada por um quadrado com um número no seu interior, sendo esse o número da etapa. As etapas iniciais são indicadas com dois quadrados em vez de um, e durante a evolução do sistema uma etapa ativa é representada com um ponto dentro do quadrado.

3.1.1.2. Transições

Uma transição representa a possível evolução de uma etapa para outra. Uma transição é representada por um traço horizontal na linha que liga uma etapa a outra, e contém sempre uma condição lógica que pode depender de variáveis de entrada e/ou de variáveis internas. Apenas quando a condição lógica tem o valor verdadeiro e a etapa a montante está ativa se dá a transição. Uma transição deve ser sempre interpretada da seguinte maneira: desativação da etapa a montante, realização da ação vinculada à transição (se existir) e ativação da etapa a jusante.

3.1.1.3. Ações

Uma ação representa uma operação executada numa variável de saída. As ações podem ser contínuas ou acionadas por eventos, e representam-se por retângulos ligados à etapa associada, dentro dos quais está a expressão representativa da ação em questão.

As ações contínuas são continuamente executadas quando as etapas às quais estão associadas estão ativas e enquanto a condição de ativação for verdadeira. Se não houver condição de ativação, esta é considerada como sendo sempre verdadeira, e a ação só depende das etapas às quais está associada. Este tipo de ações não pode ser utilizado associado a transições, pois isso significaria que a ação era realizada continuamente durante um instante, e isso não faz sentido.

As ações acionadas por eventos são executadas uma vez sempre que uma das etapas às quais estão associadas está ativa e ocorre o evento em causa. Estas ações são típicas na alteração de variáveis não booleanas, por exemplo incrementar um contador.

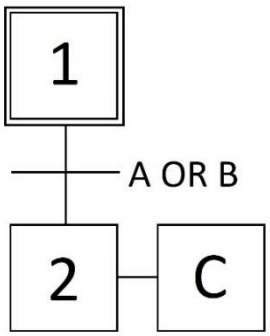

Figura 24 - Exemplo elementar de um Grafcet.

3.2. Conversão de um grafcet em código para PLC

O modelo funcional de um sistema automático é útil em diversos aspetos mas, em última análise, o objetivo não é ter um modelo, mas sim um controlador que imponha ao sistema alvo o comportamento descrito por esse modelo. Portanto, a geração do código para o controlador a partir do modelo é claramente um tema de elevada importância.

A transformação de GRAFCET em código para PLC é um assunto que é, ainda hoje, alvo de pesquisa por parte da comunidade académica, havendo diversas formas de o fazer, para várias linguagens de programação, cada uma com as suas vantagens e desvantagens [10, 11]. Muitos destes métodos são deveras úteis na geração automática de código, mas nem sempre são os mais eficientes ou fáceis de ler, dificultando o diagnóstico. Nesta dissertação foi usado um método desenvolvido pelo autor que, não sendo tão robusto como os propostos nos artigos científicos pesquisados, é de fácil implementação e leitura.

3.2.1. Programação síncrona vs. assíncrona

Antes de começar a explicação do método utilizado para converter GRAFCET em código de PLC, é conveniente deixar uma breve nota sobre os tipos de programação síncrona e assíncrona.

Para tal, tome-se como exemplo uma situação em que as condições de transição sejam verdadeiras para duas etapas consecutivas, diga-se da etapa 1 para a etapa 2 e da etapa 2 para a etapa 3, podem acontecer duas coisas a nível do processamento do código no PLC:

- O PLC ativa a etapa 2 e não ativa a 3 no primeiro ciclo, e desativa a etapa 2 e ativa a 3 no segundo ciclo;
- O PLC ativa a etapa 2, desativa-a e ativa a 3 no mesmo ciclo.

O primeiro é um caso de programação dita síncrona, onde o PLC verifica todas as condições de transição antes de efetuar qualquer uma. Como para poder haver uma transição é preciso não só que se cumpra a condição lógica da transição, mas também que a etapa anterior a essa transição esteja ativa, o PLC só vai considerar válida a transição da, no exemplo dado, etapa 1 para etapa 2. Na programação síncrona o código está escrito

de maneira a que o PLC tenha primeiro de verificar todas as transições, mudar os estados de acordo com as transições e mudar as variáveis de saída de acordo com os estados

No segundo caso está-se perante um tipo de programação assíncrona. Este tipo de programação acarreta alguns riscos, mas conduz a um maior desempenho, na medida em que o programa demora menos tempo a correr. Segundo este método o programa está escrito de maneira a que o PLC comute as etapas assim que verifique uma transição verdadeira, poupando tempo visto que não tem de reescrever todas as variáveis de estado em todos os ciclos. Um dos riscos de programar desta forma é o aparecimento de etapas fugazes, como no exemplo, em que a etapa 2 é ativada e desativada no mesmo ciclo. Isto não tem necessariamente de ser um problema, se o programador estiver ciente do risco, e pode até ser útil em alguns casos.

3.2.2. Método de conversão

Como já foi mencionado, a linguagem de programação de PLC's mais utilizada neste trabalho foi a de texto estruturado e, portanto, o método de conversão apresentado é de GRAFCET para texto estruturado. O código resultante encaixa-se na categoria de "programação assíncrona". Este método de conversão não foi alvo de estudo extensivo, visto não ser o objetivo final da dissertação, mas sim corrigido e aperfeiçoado à medida que foram descobertas falhas ou que houve necessidade de implementar novas funções. O método não prevê a implementação das estruturas hierárquicas previstas no GRAFCET pois estas nunca foram usadas neste trabalho.

A melhor forma de mostrar como esta conversão se processa é com um exemplo. Assim sendo, tome-se o grafcet da Figura 25.

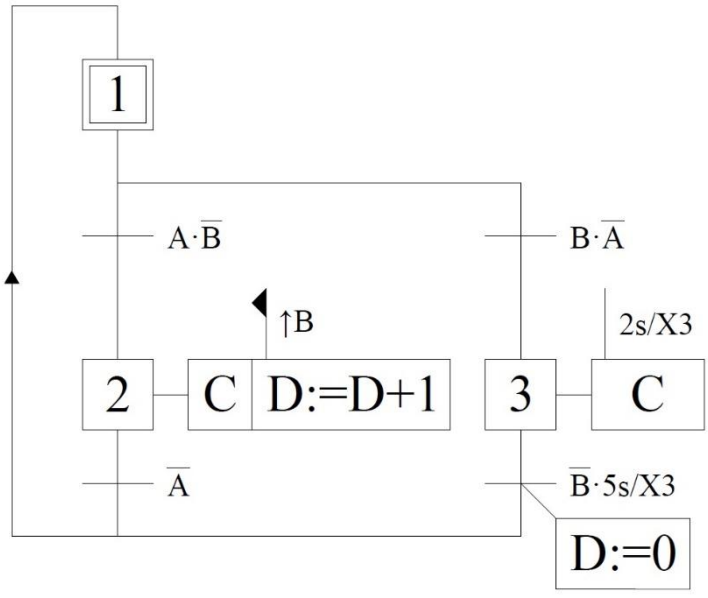

Figura 25 - Grafcet de exemplo para converter em código para PLC.

Este grafcet simples contém a maior parte das situações presentes nos Grafcets desenvolvidos para controlo ao longo desta dissertação, sendo estas:

- Seleção de transição (casos em que há duas ou mais transições possíveis);
- Ações contínuas sem condição;

- Ações contínuas com condição;
- Ações acionadas por eventos.

Esta caso apresenta três variáveis internas, que são as variáveis booleanas para cada etapa que definem se essa etapa está ativa, duas variáveis de entrada booleanas, A e B, e duas variáveis de saída, C e D, booleana e inteira, respetivamente.

O primeiro passo na programação em texto estruturado é a inicialização das variáveis internas ao programa. Se se denominar as variáveis das etapas por X, então X1 terá de ser inicializada como verdadeira, porque a etapa 1 é a etapa inicial, ou seja, está ativa no instante t=0, e X2 e X3 terão naturalmente de ser inicializadas como falsas. Além disto, como existem duas temporizações e uma transição ascendente, é preciso inicializar dois temporizadores *on-delay* e um detetor de transições ascendentes. O código resultante está apresentado na Figura 26.

```
1  PROGRAM PLC_PRG
2  VAR
3      //Etapas
4      X1: BOOL:= TRUE;
5      X2: BOOL:= FALSE;
6      X3: BOOL:= FALSE;
7      //Temporizadores
8      timer1: TON;
9      timer2: TON;
10     //Detetor de transição ascendente
11     trigger1: R_TRIG;
12 END_VAR
13 VAR_INPUT
14     A: BOOL;
15     B: BOOL;
16 END_VAR
17 VAR_OUTPUT
18     C: BOOL;
19     D: INT;
20 END_VAR
```

Figura 26 - Inicialização das variáveis para o grafcet de exemplo.

Após a inicialização das variáveis, vem o código do programa propriamente dito. O método utilizado propõe então que, em primeiro lugar, se corram as linhas de código referentes aos temporizadores e detetores de transições, de forma a que as saídas destes possam ser usadas coerentemente ao longo do programa.

Em seguida verifica-se, para uma etapa de cada vez, se esta está ativa e, se estiver, se os eventos que condicionam ações ligadas a esta etapa são dados como verdadeiros. Caso sejam, realizam-se imediatamente as ações. Também se verifica ainda se as condições de transição são verdadeiras, e caso sejam, as variáveis das etapas que se ativam e desativam tomam os valores correspondentes.

Por fim escrevem-se as variáveis das ações contínuas com e sem condição. A Figura 27 apresenta o código para o grafcet de exemplo e notas auxiliares à compreensão do que foi explicado nos últimos parágrafos.

```
1  //Temporizadores
2  timer1(IN:=X3,PT:=T#2S);
3  timer2(IN:=X3,PT:=T#5S);
4  //Detetores de transição
5  trigger1(CLK:=B);
6
7  //Etapa 1
8  IF X1 AND A AND NOT B THEN
9      X1:=FALSE;
10     X2:=TRUE;
11 ELSIF X1 AND B AND NOT A THEN
12     X1:=FALSE;
13     X3:=TRUE;
14 END_IF
15
16 //Etapa 2
17 IF X2 THEN
18     IF trigger1.Q THEN
19         D:=D+1;
20     END_IF
21     IF NOT A THEN
22         X2:=FALSE;
23         X1:=TRUE;
24     END_IF
25 END_IF
26
27 //Etapa 3
28 IF X3 AND NOT B AND timer2.Q THEN
29     D:=0;
30     X3:=FALSE;
31     X1:=TRUE;
32 END_IF
33
34 //Ações contínuas
35 C:= X2 OR X3 AND timer1.Q;
```

Temporizadores e detetores de transição no início do programa.

Verificação das condições de transição da etapa 1 para a etapa 2 e para a etapa 3.

Deteção do evento "transição ascendente da variável B" e realização da ação correspondente.

Realização da ação associada à transição da etapa 3 para a etapa 1.

Escrita da variável C que está presente tanto numa ação contínua sem condição como numa com condição.

Figura 27 - Código de PLC em texto estruturado para o grafcet de exemplo.

4. Estudo de casos de ordenação – cenários sintéticos

De maneira a, em primeira instância, adquirir maior experiência com o *software* de programação Codesys, foram inicialmente construídos, estudados e controlados cenários virtuais bastante simples que rapidamente evoluíram para outros bem mais complexos. No entanto, embora simples e clássicos em termos dos componentes industriais simulados, o propósito destes cenários era, contudo, abordar e familiarizar o estudante com uma questão importante no meio industrial e raras vezes introduzida no ensino da engenharia de automação: a sequenciação de componentes de acordo com determinados critérios.

Este tema é particularmente interessante por três razões: primeiro porque requer a busca de algoritmos de tomada de decisão flexíveis e adequados; em segundo porque requer a materialização desses algoritmos em modelos de movimentos de equipamentos industriais diversos e cooperantes; em terceiro, há a transposição dos algoritmos de decisão e dos modelos de controlo para programas de PLC, que necessariamente terão de ser executados em tempo real e, eventualmente, em controladores distribuídos.

No sentido de criar cenários potencialmente representativos da realidade industrial, e após alguma consideração, concluiu-se que a ordenação de objetos disponíveis no Factory I/O seria uma possibilidade de adequação inquestionável ao contexto pretendido. Embora implementados no contexto deste simulador, os algoritmos de tomada de decisão criados têm, no entanto, como objetivo servir a ordenação de objetos segundo um qualquer atributo inequivocamente qualificável ou quantificável, por exemplo: massa; volume, cor; etc…

Os cenários criados permitem a aplicação de dois algoritmos de tomada de decisão: um para aplicações de ordenação vertical de peças e outro para ordenação na horizontal. Desse estudo resultou um cenário onde se faz a ordenação horizontal e dois onde se faz a ordenação vertical.

4.1. Algoritmos de ordenação

Os algoritmos criados foram pensados de forma a serem o mais genéricos possível, podendo assim ser usados em cenários diferentes dos aqui relatados, desde que com objetivos semelhantes. Cada um destes algoritmos foi gerado partindo de certos princípios, que serão explicados a seguir para cada um deles, e por isso implicam que quem os queira usar em algum cenário tenha de adaptar o programa do controlador desse cenário de forma a fazer uso correto do algoritmo.

4.1.1. Ordenação horizontal

Este algoritmo parte do princípio de que existe uma série de objetos distintos alinhados de forma aleatória, identificado cada um deles por um valor. O algoritmo pretende remover dessa série o objeto cujo valor identificativo é o mais elevado de forma a criar séries sucessivas, até que as peças estejam organizadas segundo o seu valor. Até ao fim deste subcapítulo as séries serão designadas por filas, pois esta palavra relaciona-

se de forma mais intuitiva com as imagens apresentadas. Nas Figura 28 e Figura 29 exibe-se um caso exemplo.

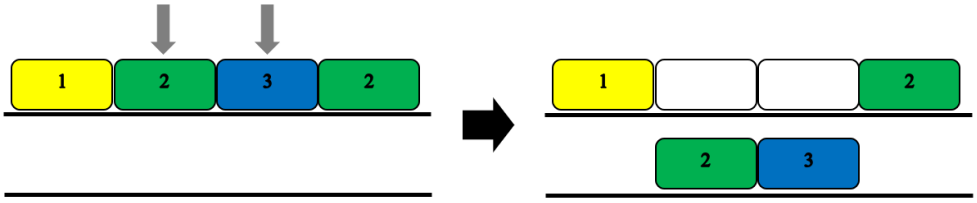

Figura 28 - Exemplo de aplicação do algoritmo de ordenação horizontal – parte 1.

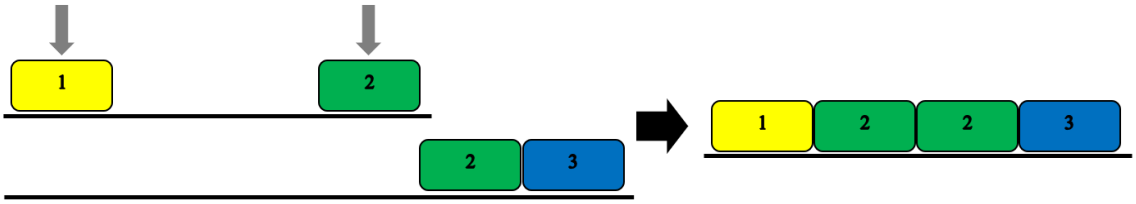

Figura 29 - Exemplo de aplicação do algoritmo de ordenação horizontal - parte 2.

Partindo do princípio que o algoritmo tem como objetivo ordenar as peças por ordem crescente da esquerda para a direita, este compreende os seguintes passos:

i. Identificar o maior valor dentre os valores de peças da 1ª fila.
ii. Remover todas as peças com esse valor.
iii. Identificar o novo maior valor dentre os valores de peças da 1ª fila.
iv. Remover todas as peças com esse valor que se encontrem à esquerda da peça mais à esquerda da 2ª fila.
v. Se todas as peças com esse valor puderem ser removidas, voltar ao ponto iii, caso contrário ir para o ponto vi.
vi. "Esvaziar" a 2ª fila e, caso ainda haja peças na 1ª fila, voltar ao ponto i.

No caso do exemplo, após remover a peça de maior valor, 3, ainda foi possível remover uma das peças de valor 2, mas a outra peça de valor 2 não poderia ser simultaneamente removida pois violar-se-ia a ordem. Assim sendo, foi necessário "esvaziar" a 2ª fila e voltar ao primeiro passo.

Por forma a ser utilizado num programa de PLC, este algoritmo teve de ser materializado numa função, à qual se fornece dados sobre a situação atual (*inputs*) a partir dos quais esta determina as peças a serem movidas (*outputs*). A função gerada para aplicação do algoritmo não devolve todos os passos desde o início até ao fim do algoritmo, mas sim apenas a informação de quais as próximas peças a passar da 1ª para a 2ª fila. Desta forma o programa de controlo tem mais flexibilidade no uso da função. Em termos de código, na Tabela 1 estão apresentadas quais as variáveis de entrada e saída da função.

Tabela 1: Variáveis de entrada e de saída da função referente ao algoritmo de ordenação horizontal.

Entradas	Saídas	Tipo	Descrição
rowAin		Array of Real	*Array* de valores das peças na 1ª fila antes da manipulação
	rowAout	Array of Real	*Array* de valores das peças na 1ª fila após manipulação
	rowBout	Array of Real	*Array* de valores das peças na 2ª fila após manipulação
	moves	Array of Bool	*Array* indicativo das peças que vão ser manipuladas
	nummoves	Int	Número de peças manipuladas

Como nota, diga-se que o algoritmo de ordenação horizontal é assim designado por ter sido pensado para ser usado num cenário de ordenação horizontal de objetos. No entanto este algoritmo pode ser utilizado em qualquer situação que seja análoga à situação que serviu de ponto de partida ao seu desenvolvimento.

4.1.2. Ordenação vertical

De modo semelhante ao anterior, este algoritmo parte do princípio que existe uma série de peças dispostas aleatoriamente e que estas são identificadas por um valor numérico. No entanto, neste caso, o algoritmo considera que, de cada série de peças, só a de um dos extremos pode ser movida, exatamente como se fosse uma pilha de caixas da qual só se poderia mover uma de cada vez, e tendo sempre de ser a do topo. Ora, esta limitação na manipulação das peças obriga à existência de mais do que duas séries de caixas, de forma a que uma deles possa ser utilizada como *buffer*. Até ao fim deste subcapítulo as séries serão designadas por pilhas (como em pilhas de caixas), já que esta palavra se relaciona de forma mais intuitiva com as imagens a seguir apresentadas. A Figura 30 apresenta um esquema de uma possível situação inicial e dos movimentos que o algoritmo pode realizar. A pilha final que contém as peças ordenadas nunca poderá ser a 1ª, pois movimentos da 2ª ou 3ª pilhas para a 1ª não são possíveis, de acordo com o esquema.

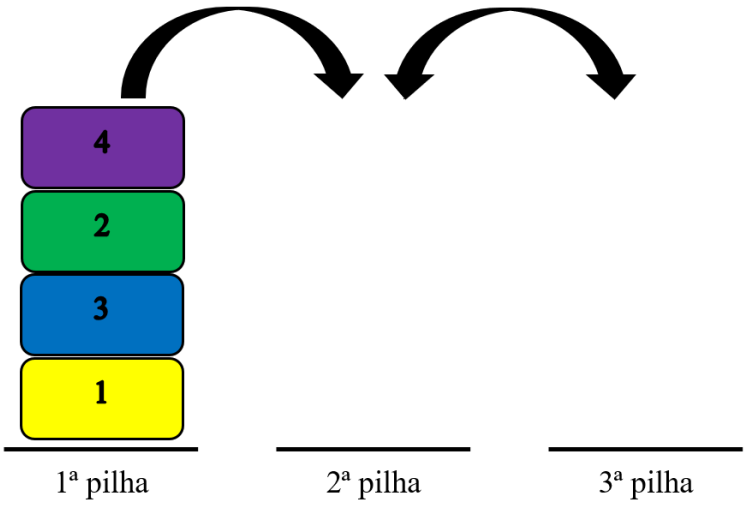

Figura 30 - Situação inicial do algoritmo de ordenação vertical.

A ideia que simplifica a ordenação das peças é a de que, mantendo uma ordem crescente de peças numa pilha e uma ordem decrescente de peças na outra, e tendo uma das pilhas apenas peças de maior valor do que a outra (como na Figura 31), é sempre possível mover peças entre as pilhas de forma a que uma nova peça possa ser inserida sem comprometer a ordem geral.

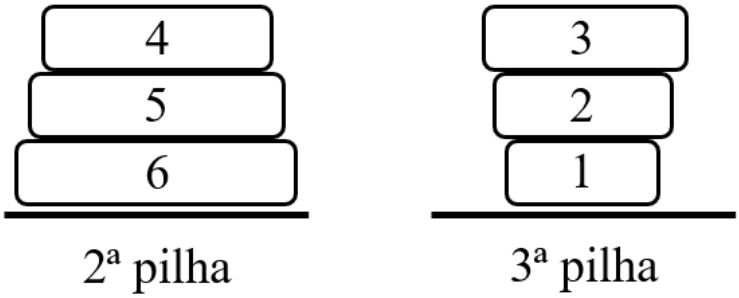

Figura 31 - Princípio base do algoritmo de ordenação vertical.

Se se mantiver a ordem como no esquema, no final basta passar as peças de menor valor para a pilha que contém as peças de maior valor para se obter uma pilha ordenada. Tanto a 2ª como a 3ª pilhas podem se tornar na pilha de saída, tornando-se a outra a pilha de apoio (*buffer*). Assim que as duas pilhas tenham pelo menos uma peça é possível determinar qual a pilha que será a de saída, pois essa será sempre a que contêm as peças de maior valor.

Sabendo-se isto, pode agora ser explicada a lógica do algoritmo no seu todo.

i. Movimentar a peça do topo da pilha de entrada (1ª pilha) para a 2ª pilha.
ii. Se a próxima peça na pilha de entrada for de igual valor à primeira peça movimentada, então movimentar esta peça para a 2ª pilha também, caso contrário, movimentar a peça para a 3ª pilha. A partir deste momento já se sabe qual é a pilha de saída.
iii. Caso haja peças na 1ª pilha, ir para o ponto iv, caso contrário ir para o ponto vi.
iv. Se o valor da peça no extremo da 1ª pilha se encontrar entre os valores das peças nos topos da 2ª e 3ª pilhas, então movimentar a peça da 1ª para a 2ª pilha. Caso contrário, ir para o ponto v.
v. Se o valor da peça no topo da 1ª pilha for maior que os valores das peças nos topos da 2ª e 3ª pilhas, movimentar a peça no topo da pilha de saída para a pilha de apoio. Se o valor da peça no topo da 1ª pilha for menor que os valores das peças nos topos da 2ª e 3ª pilhas, movimentar a peça no topo da pilha de apoio para a pilha de saída. Em seguida voltar ao ponto iii.
vi. Movimentar todas a peças da pilha de apoio para a pilha de saída.

No caso do exemplo, a primeira peça a ser movida é, naturalmente, a peça 4, para a 2ª pilha, e de seguida a peça 2, para a 3ª pilha (passando pela 2ª antes), ficando a situação como mostra a Figura 32. Neste momento sabe-se já que a pilha de saída vai ser a 2ª, pois esta tem a peça de maior valor.

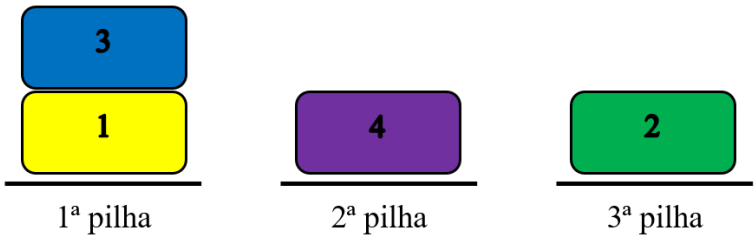

Figura 32 - Exemplo de aplicação do algoritmo de ordenação vertical - parte 1.

Nesta situação a próxima peça da 1ª pilha, peça 3, pode perfeitamente ser movida para a 2ª pilha sem estragar a ordem (Figura 33).

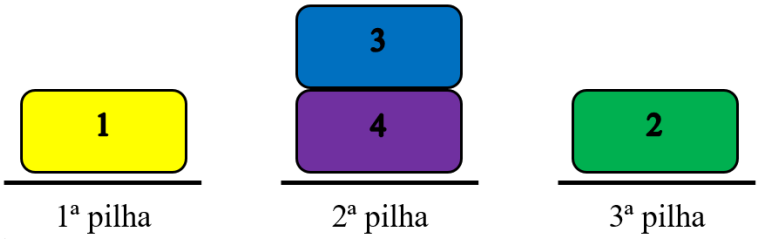

Figura 33 - Exemplo de aplicação do algoritmo de ordenação vertical - parte 2.

Já na situação da Figura 33, a peça 1 não pode ser movida, pois isso impediria a formação de uma pilha ordenada. É preciso então passar a peça 2 para a 2ª pilha, e posteriormente passar a peça 1 para a 2ª pilha, fazendo assim uma pilha ordenada, como se pode ver na Figura 34.

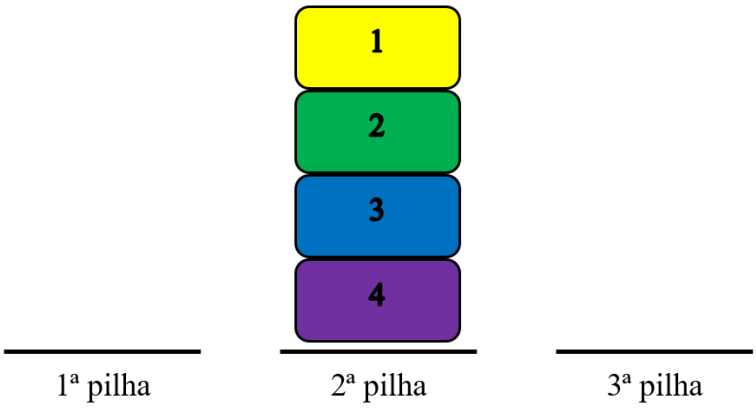

Figura 34 - Exemplo de aplicação do algoritmo de ordenação vertical - parte 3.

O que nunca pode acontecer é acabar com a pilha final no lugar da 1ª pilha, isto porque o algoritmo não prevê movimentos da 2ª ou 3ª pilhas para a 1ª (de acordo com o esquema da Figura 30).

Da mesma maneira que a função criada para a aplicação do algoritmo de ordenação horizontal, a função criada para aplicação deste algoritmo não devolve todos os passos desde o início até ao fim, mas apenas informa de qual o próximo movimento a realizar. Neste caso a vantagem ainda é maior, pois os movimentos a realizar só dependem da peça seguinte a retirar da 1ª pilha e das peças presentes na 2ª e 3ª pilhas e, assim sendo, é possível modificar a 1ª pilha durante o processo de ordenação (por exemplo adicionando-lhe peças) sem comprometer a ordem da pilha final. Em termos de código, na Tabela 2 estão apresentadas quais as variáveis de entrada e saída da função.

Tabela 2: Variáveis de entrada e saída da função referente ao algoritmo de ordenação vertical.

Entradas	Saídas	Tipo	Descrição
stackAin		Array of Real	*Array* de valores das peças na 1ª pilha antes da manipulação
stackBin		Array of Real	*Array* de valores das peças na 2ª pilha antes da manipulação
stackCin		Array of Real	*Array* de valores das peças na 2ª pilha antes da manipulação
	stackAout	Array of Real	*Array* de valores das peças na 1ª pilha após manipulação
	stackBout	Array of Real	*Array* de valores das peças na 2ª pilha após manipulação
	stackCout	Array of Real	*Array* de valores das peças na 3ª pilha após manipulação
	moveA_B	Bool	Indicação para realizar movimento da 1ª para a 2ª pilha
	moveB_C	Bool	Indicação para realizar movimento da 2ª para a 3ª pilha
	moveC_B	Bool	Indicação para realizar movimento da 3ª para a 2ª pilha
	stackBheavy	Bool	Indicação se a 2ª pilha é a pilha com as peças de valor maior

Como nota, diga-se que o algoritmo de ordenação vertical, tal como no de ordenação horizontal, é assim designado por ter sido pensado para e usado em cenários de ordenação vertical de objetos. No entanto este algoritmo pode ser utilizado em qualquer situação que seja análoga à situação que serviu de ponto de partida ao seu desenvolvimento.

4.2. Cenário de ordenação horizontal

De forma a aplicar o algoritmo de ordenação desenvolvido a uma possível situação industrial foi criado um cenário virtual no Factory I/O que fosse adequado a tal. Neste cenário, os objetos a serem ordenados são discriminados por massa. A cada massa está associada uma cor, de forma a ser possível ver sem dificuldade se a ordenação ocorre corretamente. Este cenário pode ser visto na Figura 35.

4.2.1. Elementos utilizados

O cenário é composto por um conjunto de elementos, cujas funções são descritas a seguir.

- A balança, no topo da qual são emitidas as caixas, uma a uma;
- O tapete 1, responsável por levar as caixas da balança até ao *pusher* respetivo;
- Os *pushers*, que movem as caixas do tapete 1 para o tapete 2;
- Os sensores, que são utilizados de forma a se saber quando se pode ativar cada lâmina de paragem;
- As lâminas de paragem, que isolam as caixas nos seus espaços respetivos;
- O tapete 2, que desloca as caixas para desimpedir o espaço em frente aos *pushers*;
- O tapete 3, que acumula as caixas de forma compacta, antes de as enviar para remoção.

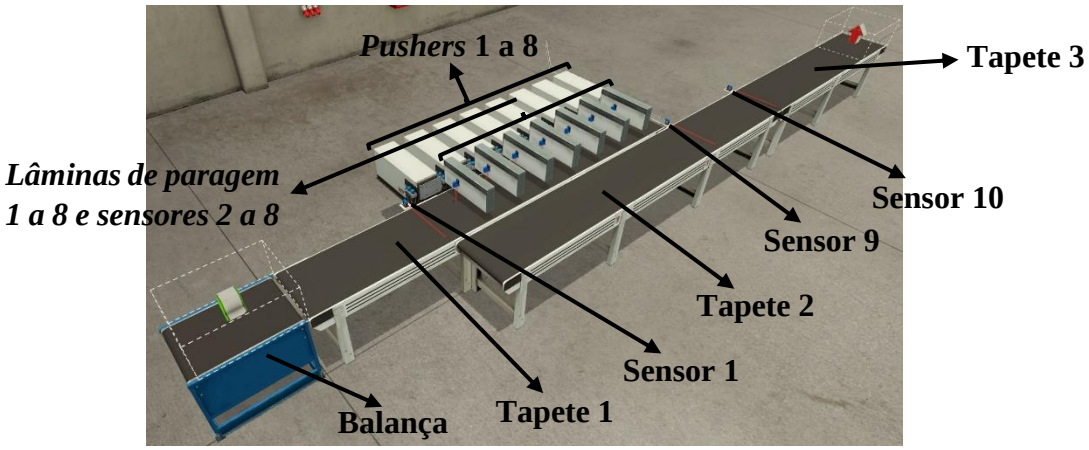

Figura 35 - Cenário virtual para ordenação horizontal.

4.2.1. Descrição do funcionamento

O cenário envolve uma balança, onde caixas de paletização são emitidas e pesadas, sendo as respetivas massas registados num vetor por ordem de entrada em cena. Em seguida, as caixas são direcionadas para compartimentos separados, cada um com um *pusher*. Aplicando o algoritmo de ordenação horizontal, descobre-se quais são os *pushers* que devem ser ativados, e procede-se com essa ativação, deslocando em seguida as peças que foram empurradas de forma a desimpedir a zona do tapete 2 em frente aos *pushers*. Este processo repete-se até todas as peças terem sido empurradas, sendo que estas são então enviadas, já ordenadamente, para um eliminador. A Figura 36 mostra o cenário em situação de funcionamento.

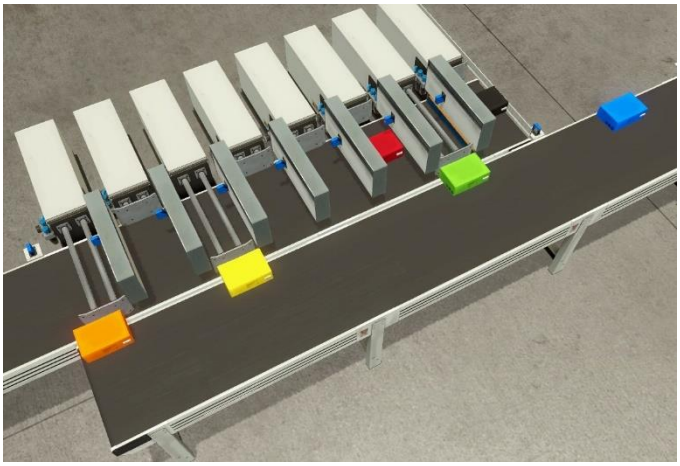

Figura 36 - Cenário virtual para ordenação horizontal em funcionamento.

O código de controlo deste cenário, e dos outros expostos a seguir, pode ser analisado e, a partir dele, criarem-se grafcets. Estes grafcets facilitam imenso a compreensão do código pois combinam a complexidade inerente á linguagem de programação utilizada com a simplicidade que advém de um modelo em GRAFCET. No anexo A encontram-se os grafcets produzidos a partir do código de controlo deste cenário.

4.3. Cenário de ordenação vertical com dois manipuladores cartesianos

O algoritmo de ordenação vertical foi explorado em dois cenários diferentes, ilustrando assim a sua aplicabilidade a múltiplas situações e recursos industriais. O primeiro desses cenários, retratado na presente secção, faz uso de dois manipuladores cartesianos e, de certa forma, decalca muito diretamente os esquemas apresentados aquando da explicação do algoritmo. Este cenário pode ser observado na Figura 37.

4.3.1. Elementos utilizados

O cenário é composto por vários dispositivos, cada um com um propósito específico, descrito a seguir.

- A balança, em cima da qual a pilha de caixas é emitida;
- O tapete 1, que transporta a pilha inicial para os manipuladores;
- Os manipuladores, cuja propósito é transformar a pilha inicial desordenada numa pilha final ordenada;
- O tapete 2, que transporta a pilha final para eliminação.

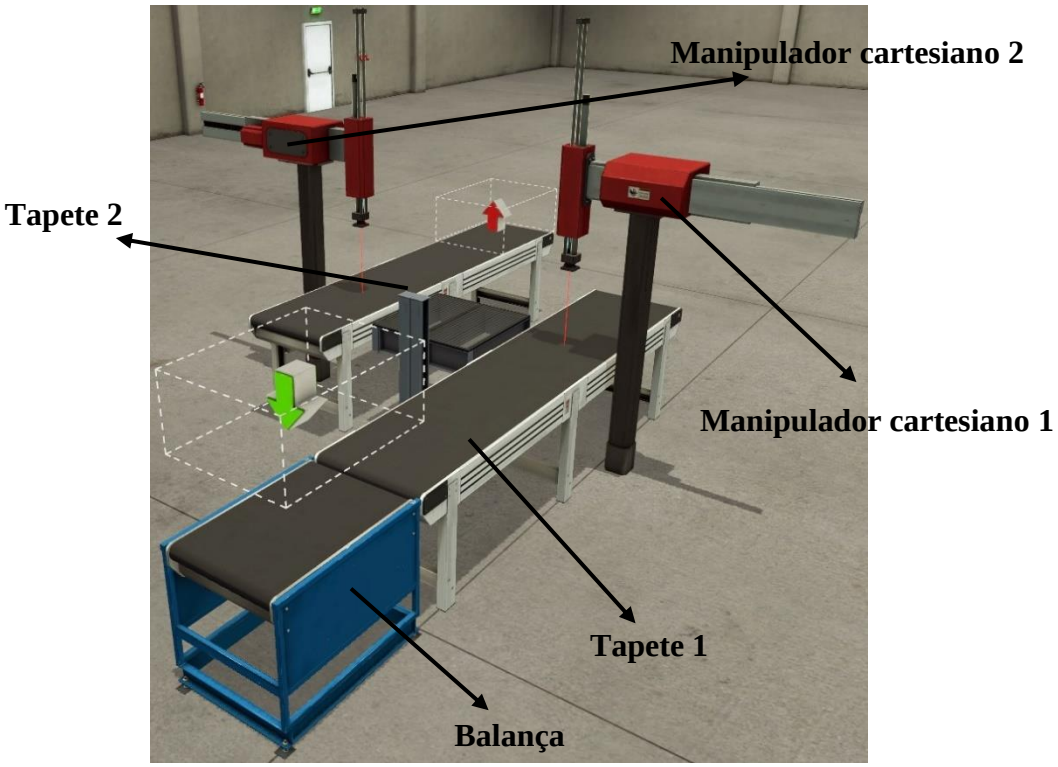

Figura 37 - Cenário virtual para ordenação vertical utilizando dois manipuladores cartesianos.

4.3.2. Descrição do funcionamento

De forma semelhante ao cenário de ordenação horizontal, este envolve também uma balança onde caixas de paletização são emitidas e pesadas. No entanto, neste caso as caixas são emitidas umas em cima das outras, formando uma pilha, e as massas das caixas são registadas num vetor. Em seguida, a pilha de caixas é encaminhada por um tapete para um manipulador cartesiano que, juntamente com um segundo manipulador, constrói uma pilha ordenada. O primeiro manipulador move caixas apenas da 1ª para a 2ª pilha, enquanto que o segundo manipulador as move entre as 2ª e 3ª pilhas. No final, a pilha

ordenada é encaminhada até ao eliminador. Uma imagem do cenário em funcionamento encontra-se na Figura 38.

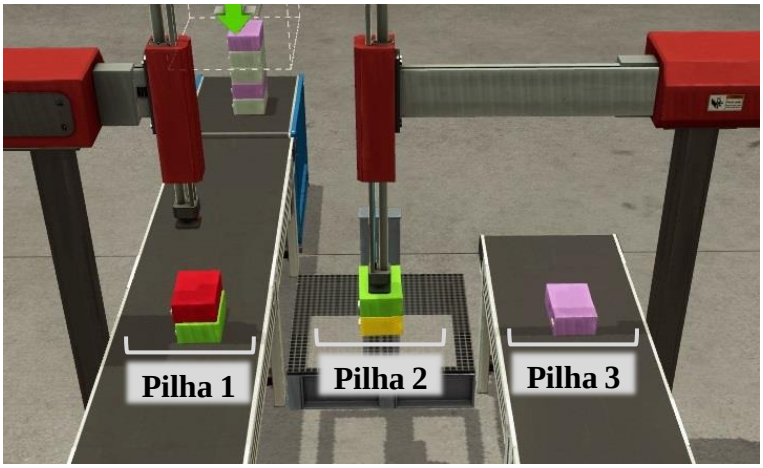

Figura 38 - Cenário virtual para ordenação vertical com dois manipuladores cartesianos em funcionamento.

Como foi dito aquando da explicação do algoritmo, no final do processo a pilha ordenada pode ser tanto a 2ª como a 3ª, ou seja, no cenário em questão a pilha ordenada poderia tanto acabar em cima do tapete de saída ou em cima da estrutura que está entre os tapetes, se nada fosse feito em contrário É, portanto, preciso haver o cuidado de programar o controlador de maneira a evitar que tal aconteça.

4.4. Cenário de ordenação vertical com um manipulador cartesiano

O segundo cenário no qual o algoritmo de ordenação vertical foi usado não é um exemplo tão semelhante ao esquema exibido na explicação do algoritmo, mas é, não obstante, um caso de aplicação. Este cenário pode ser visto na Figura 39.

4.4.1. Elementos utilizados

O cenário é composto por vários elementos, embora menos que o anterior, cujos propósitos são descritos a seguir.

- A balança, em cima da qual a pilha de caixas é emitida;
- O tapete 1, que transporta a pilha inicial para o manipulador;
- O manipulador que, em conjunto com o tapete 2, reorganiza as caixas da pilha inicial numa pilha final ordenada;
- O tapete 2 que, para além de enviar a pilha final para eliminação, também é crucial no processo de reordenação das caixas.

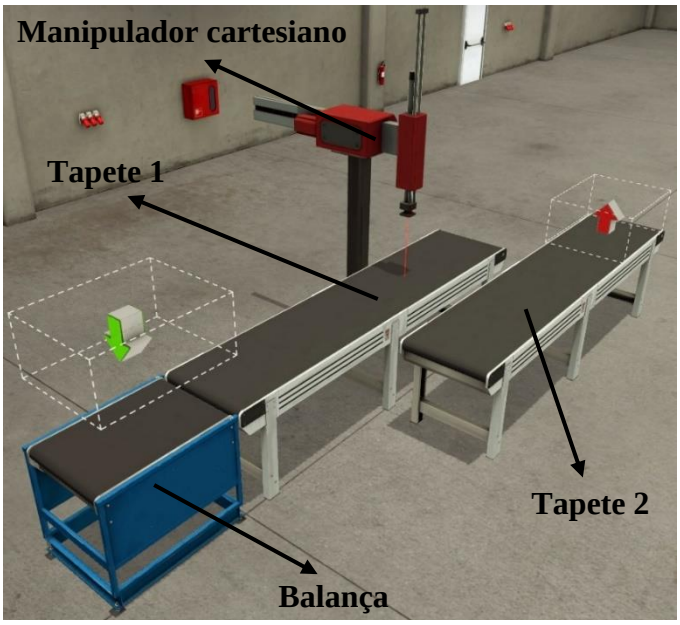

Figura 39 - Cenário virtual para ordenação vertical utilizando um manipulador cartesiano.

4.4.2. Descrição do funcionamento

Tal como nos outros casos já mencionados, neste também se utiliza a massa para distinguir as caixas que pretende ordenar, e as estas chegam ao manipulador numa pilha. Após ordenação numa nova pilha sobre o tapete de saída, esta é encaminhada para o eliminador. Neste caso pode parecer que não há lugar a uma terceira pilha, sendo que só há dois tapetes e um manipulador, mas o segundo tapete pode mover-se e alterar a pilha na qual o manipulador vai pousar ou retirar peças. Uma imagem do cenário em funcionamento encontra-se na Figura 40.

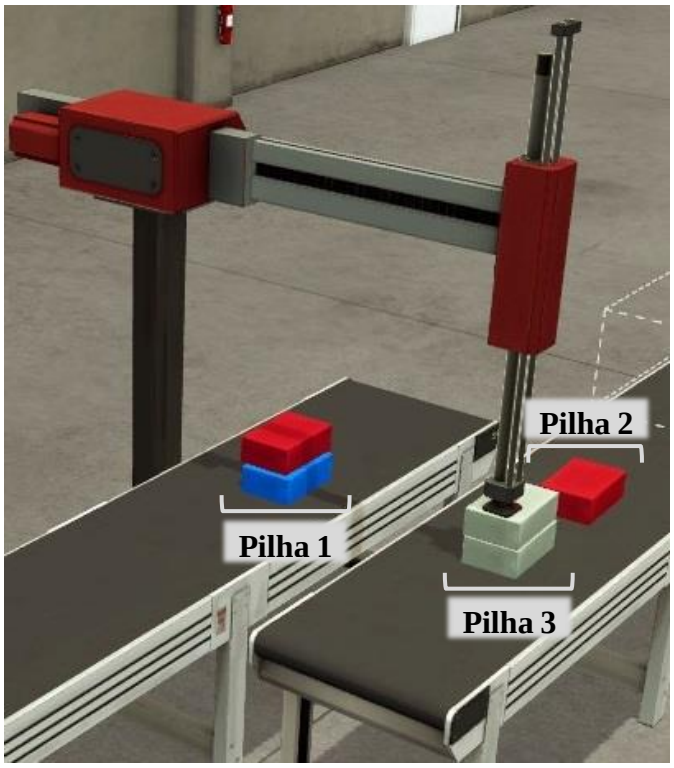

Figura 40 - Cenário virtual para ordenação vertical com um manipulador cartesiano em funcionamento.

Por natureza este mecanismo permite realizar um movimento que não está previsto no algoritmo, que é a movimentação direta de caixas da 1ª para a 3ª pilha, e que pode ser usado para diminuir o tempo de ordenação.

5. Estudo de casos de controlo de movimento – cenário híbrido

A utilização de componentes motrizes, em especial de acionamentos elétricos, em contextos industriais é muitíssimo comum e nem sempre trivial, pelo que a capacidade de controlar a dinâmica desses componentes é demasiado importante para ser descurada no ensino da automação. Assim, o presente capítulo retrata um cenário adequado à utilização e controlo deste tipo de componentes.

A situação mencionada consiste num cenário industrial virtual, no qual estão presentes, para além de equipamentos de transporte e manipulação de objetos já utilizados na aplicação relatada no anterior capítulo, dois centros de maquinagem pelos quais passam objetos para serem transformados de acordo com parâmetros definidos no PLC. Estes centros de maquinagem são usados como "ponte" entre a simulação e a realidade, na medida em que, enquanto estes operam na simulação, há motores elétricos reais que entram em funcionamento como se dos acionamentos das máquinas se tratasse. Dependendo do objeto a ser transformado, os motores seguem um perfil de velocidade predefinido. Como os motores precisam de uma forma de se ligarem ao PLC, é necessária a utilização de outros equipamentos e funções ligados ao controlo de movimento, nomeadamente variadores de frequência, *drives* e funções especiais dos PLC's.

Além disto, de forma a explorar também o processo de criação de interfaces para o utilizador, foi criada uma HMI muito flexível que permite a definição e modificação de parâmetros do PLC, como perfis de velocidade para os motores ou número de peças a ser produzidas, e visualização de dados em tempo real, como a velocidade de cada motor ou o número de peças já produzidas.

5.1. Cenário industrial virtual

O cenário industrial em questão reproduz um sistema que consiste na produção e posterior separação de peças. O sistema admite peças ditas "em bruto", que chegam aos centros de maquinagem e são transformadas em peças acabadas, sendo estas depois enviadas para separação.

As peças em bruto podem ser, unicamente por limitação da versão do *software* de simulação, azuis ou verdes, e as peças acabadas podem ser tampas ou bases (também elas azuis ou verdes). A escolha das peças que são produzidas é efetuada por um utilizador através de uma interface gráfica, na qual define um lote com o número de peças que entender de cada tipo, ou quantidades aleatórias. O tipo de produção (aleatória ou em série) e a composição de cada lote (para produção em série) são definidos na HMI, e os menus para tais objetivos podem ser vistos na Figura 41.

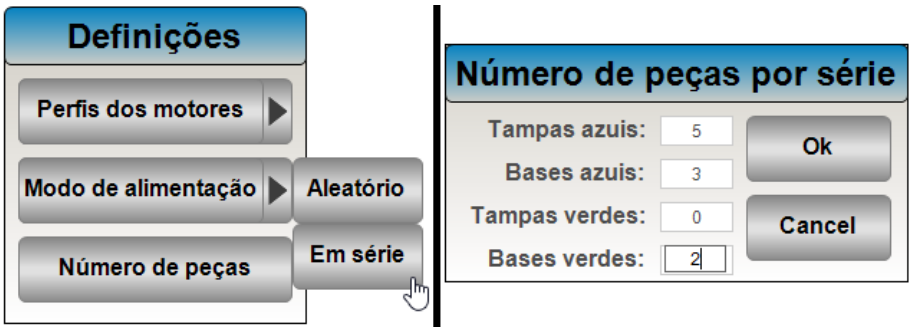

Figura 41 - Utilização da HMI para seleção do modo de alimentação e do número de peças por série.

Cada centro de maquinagem está associado ao comando de um tipo particular de motor: assim um centro é suposto de estar dotado de dois motores de indução trifásicos e o outro de dois motores passo-a-passo. Quando um destes centros entra em funcionamento os motores a ele associados devem movimentar-se segundo um perfil de velocidade, específico para o tipo de peça a ser produzida, que se inicia assim que a porta do centro de maquinagem se fecha. O movimento dos motores pretende simular o que seria o movimento de eixos pertencentes ao centro de maquinagem.

No fim da produção, a peça é encaminhada para um sistema de separação. O sistema regista, através de um sensor de visão, o tipo de peça que acabou de entrar. A peça é então encaminhada por tapetes transportadores e eventualmente desviada para um tapete específico, ao fim do qual um manipulador cartesiano se encarrega de criar uma pilha para posterior remoção. Uma imagem da parte sintética do cenário híbrido encontra-se na Figura 42, e um esquema com as nomenclaturas de cada componentes na Figura 43. A Figura 45 diz respeito à parte real do cenário, estando nela exibidos os motores associados ao centro de maquinagem 1 (trifásicos) e os motores associados ao centro de maquinagem 2 (passo-a-passo).

Figura 42 - Cenário virtual para controlo de movimento.

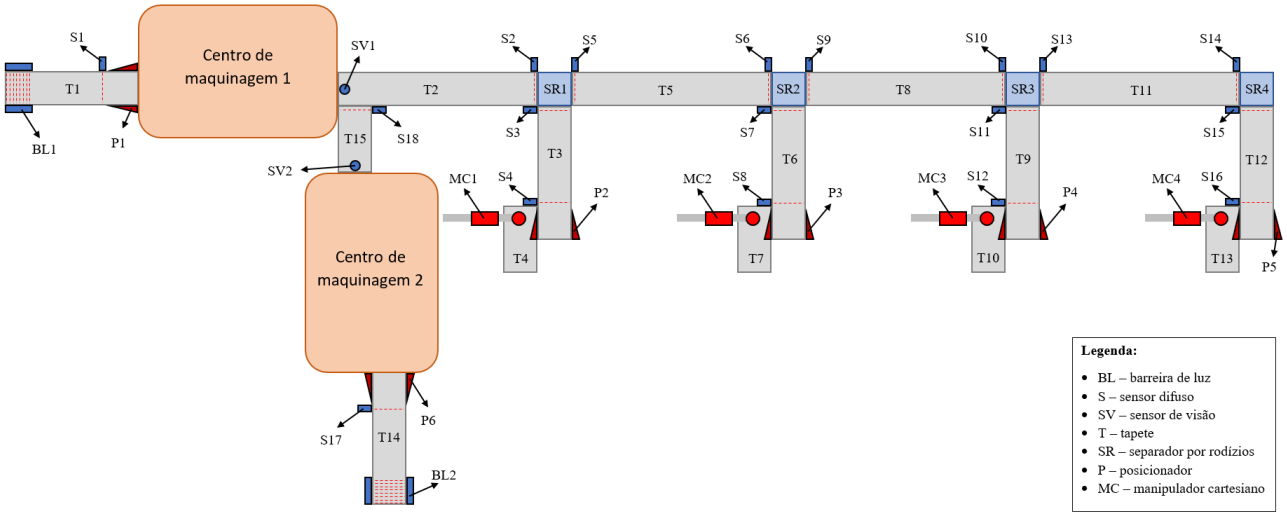

Figura 43 - Esquema do cenário virtual para controlo de movimento.

Todo este cenário foi controlado por um único controlador, o *soft-PLC* Codesys Control Win V3, cujo código de controlo foi desenvolvido e testado no ambiente de desenvolvimento integrado Codesys.

5.2. Comunicação entre cenário, controlador e motores

Para possibilitar o funcionamento simultâneo e sincronizado do cenário com os motores elétricos foi necessário estabelecer comunicação entre estes e o controlador. A estrutura resultante do tipo de comunicações escolhidas entre os diversos componentes centraliza o controlo no *soft*-PLC, estando todos os componentes ligados a ele. Um esquema destas comunicações está apresentado na Figura 46.

Havendo uma variedade de opções, o método de comunicação escolhido para efetuar a comunicação entre o Factory I/O e o *soft-PLC* foi através de um servidor OPC Codesys, visto ser eficaz e de simples utilização. Já para ligar o *soft-PLC* aos equipamentos físicos reais foi necessário um pouco mais do que escolher um protocolo de comunicação.

Em relação aos motores trifásicos, uma vez que a utilização de variadores de frequência era obrigatória para se poder controlar a sua velocidade, e os variadores de frequência disponíveis só suportavam comunicação por MODBUS RTU através de uma interface série RS422 ou RS485, não houve possibilidade de escolha no que toca ao protocolo de comunicação. Além disto, como o *soft-PLC* foi executado num computador moderno, que não possui nenhuma porta série, houve a necessidade de utilizar dois adaptadores em conjunto: um adaptador USB-RS232 e um RS232-RS485, que podem ser observados na Figura 44.

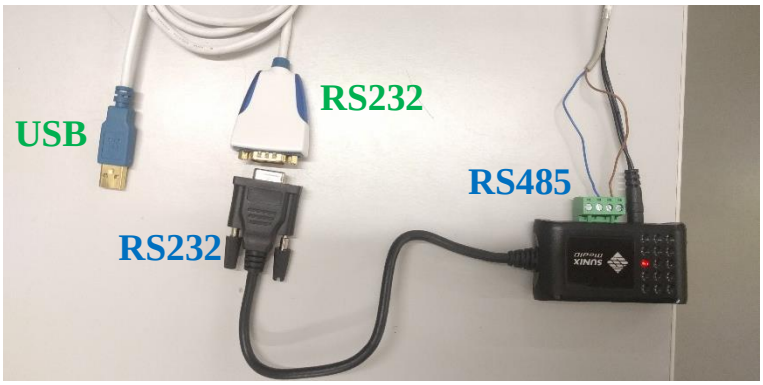

Figura 44 - Adaptadores USB-RS232 e RS232-RS485 utilizados.

No que toca aos motores passo-a-passo, a utilização de *drives* está implícita, mas os *drives* disponíveis não possuíam qualquer tipo de capacidade de comunicação, aceitando apenas sinais elétricos de I/O digital. Isto levou ao uso de um PLC, um Siemens SIMATIC S7-1200, que pode facilmente comunicar com o *soft-PLC* e tem a capacidade de enviar sinais elétricos para os *drives* dos motores. Neste caso havia bastantes protocolos possíveis de utilizar para efetuar a comunicação entre o S7-1200 e o *soft-PLC*, dos quais o escolhido foi o MODBUS TCP. Poder-se-ia ter usado o MODBUS RTU, e ter ligado o PLC na rede RS485 que já existia entre o *soft-PLC* e os variadores de frequência, ou até ligar o PLC ao servidor OPC, mas a aplicação de um protocolo diferente e moderno torna o projeto mais interessante e versátil. O PLC foi também equipado com um módulo de geração de impulsos a alta frequência (até 200kHz), pois as funções do PLC específicas para controlo de motores passo-a-passo não aceitam a utilização de saídas por relé, e o PLC em causa possui apenas esse tipo de saídas.

Na Figura 45 podem ver-se todos os componentes do cenário híbrido. Começando pela esquerda, veem-se os dois motores de indução trifásicos e a seguir os dois variadores de frequência que os alimentam. Esses variadores estão inseridos numa consola com comandos manuais que não foram utilizados. No centro está o computador usado para correr tanto o cenário virtual (visível na foto) assim como o *soft-PLC* que o controla. Pousado no computador estão os conversores que permitiram comunicar numa rede série através de uma porta USB. À direita observam-se o PLC S7-1200, os motores passo-a-passo que foram alvo de controlo, e também os componentes necessários à ligação entre o PLC e os motores, entre eles os *drives* dos motores.

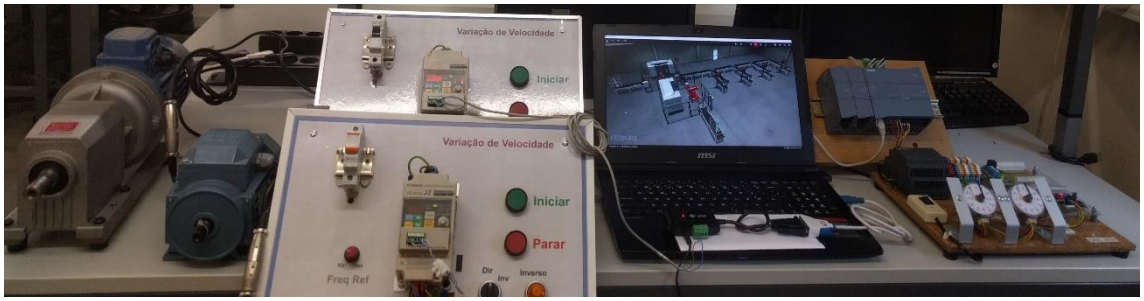

Figura 45 - Foto do cenário híbrido na sua totalidade.

Na Figura 46 estão esquematizadas as ligações entre os componentes do sistema, explicitando-se os protocolos de comunicação usados entre eles.

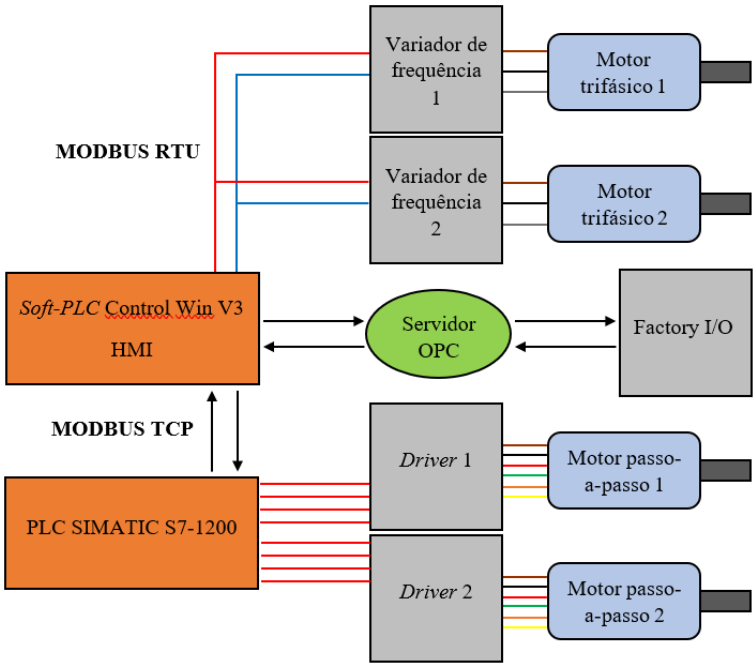

Figura 46 - Esquema das conexões e protocolos estabelecidos entre os componentes do sistema.

5.3. Controlo dos motores trifásicos

Como já foi referido, o controlo da velocidade dos motores trifásicos foi conseguido através da aplicação de variadores de frequência, variadores esses que suportavam comunicação em MODBUS RTU. Assim sendo, torna-se importante mencionar que informações é que os variadores precisam de receber para porem os respetivos motores a funcionar de acordo com o desejado, quais é que podem devolver ao *soft-PLC* de utilidade no contexto em causa e de que forma é que esta troca de informação se processa.

5.3.1. Variáveis de entrada dos variadores

Começando por referir a informação que é necessária enviar aos variadores, torna-se importante dizer que estes suportam a modificação de imensas configurações por comunicação MODBUS, das quais apenas uma pequena parte foi utilizada. A seguir estão listadas as configurações dos variadores que o PLC está programado para mudar, de acordo com o manual de instruções [12]:

- Ativação do variador (*enable*);
- Sentido de rotação;
- Frequência de referência;
- Tempo de aceleração;
- Tempo de desaceleração;
- Formato das rampas de aceleração e desaceleração.

Visto se estar a falar de comunicação em MODBUS, poder-se-ia pensar que as variáveis fossem guardadas pelos variadores como *holding registers* ou *coils*, mas o fabricante optou por utilizar apenas *holding registers*, sendo que muitas das configurações binárias são guardadas num registo deste tipo juntamente com outras. Este é o caso da ativação do variador e do sentido de rotação, que são guardadas no mesmo *holding*

register nos *bits* 0 e 1, respetivamente. As restantes configurações indicadas são números inteiros e por isso usam os 2 *bytes* do registo na sua totalidade.

As primeiras duas configurações referidas são autoexplicativas, já as restantes merecem uma breve nota. A frequência de referência é o valor que se envia ao variador e para o qual ele deve autonomamente convergir. Os tempos de aceleração e desaceleração definidos dizem respeito ao tempo que o variador leva a mudar a frequência de 0Hz até à frequência máxima definida (neste caso 100Hz), e por isso obriga à realização de cálculos por parte do *soft*-PLC quando se pretende variar a frequência entre dois valores que não esses. Quanto ao formato das rampas, este parâmetro precisa de ser definido por um valor inteiro porque, apesar de haver apenas dois formatos (em linha ou em S), o formato em S pode ser caraterizado por três tempos caraterísticos diferentes.

5.3.2. Variáveis de saída dos variadores

Tal como no caso anterior, os variadores possuem uma quantidade de variáveis de monitorização muito superior àquelas que foi necessário utilizar no presente trabalho, sendo que apenas duas variáveis são lidas pelo *soft-PLC*. Estas são:

- Estado do variador;
- Frequência atual.

O estado do variador não é sempre igual à ativação do variador, porque quando este é desativado, ou seja, a variável da ativação do variador toma o valor 0, a frequência vai diminuir segundo o tempo de desaceleração definido até 0Hz, e só aí a variável do estado do variador toma o valor 0. A frequência atual é essencial para se saber a velocidade do motor, principalmente durante as variações de frequência, onde a frequência instantânea é diferente da referência.

5.3.3. Processamento da troca de informação

Como já foi dito, os motores trifásicos estão associados a um centro de maquinagem no cenário virtual, e durante o seu funcionamento seguem um perfil de velocidade que depende da peça em transformação no cenário virtual. Dito isto, é importante então saber-se como se infere a informação a enviar aos variadores de frequência da informação sobre a peça que está a ser transformada.

Os perfis de velocidade para cada tipo de peça são guardados numa matriz com 3 colunas e número de linhas variável. Cada coluna contém uma informação diferente: tempo; frequência de referência e formato da rampa, e cada linha indica um ponto de referência no gráfico da frequência em função do tempo. O tempo é indicado, em segundos, numa escala absoluta e, portanto, tem de estar por ordem crescente à medida que se avança no índice das linhas. A primeira linha tem de conter o valor 0, porque todos os perfis se iniciam em $t = 0\ s$. A frequência pode ser um valor positivo ou negativo, sendo que um valor negativo indica que o motor gira no sentido inverso, e tanto a primeira como a última linha têm de conter o valor 0, já que o motor inicia e acaba o perfil sempre parado. A 3ª coluna indica o formato da rampa do ponto anterior para o ponto definido nessa linha e, assim sendo, o valor presente na primeira linha nunca é usado, pois esse é o ponto de partida. O número de linhas não é, na verdade, variável, porque a matriz tem

de ser definida no programa com um tamanho finito. Variável é o número de linhas utilizadas por um perfil, cujo máximo é 50, porque foi esse o número definido.

Na Figura 47 pode observar-se a tabela onde se definem os pontos do perfil de velocidade dos motores. O número de linhas indicado neste caso não é 50, mas há um botão para adicionar mais (e outro para remover) se tal for desejado. A HMI introduz um gráfico que auxilia na criação do perfil, apresentando a evolução da velocidade do motor ao longo do tempo para o perfil em edição, e que se atualiza à medida que os pontos são editados e não apenas quando se carrega no botão de "Ok".

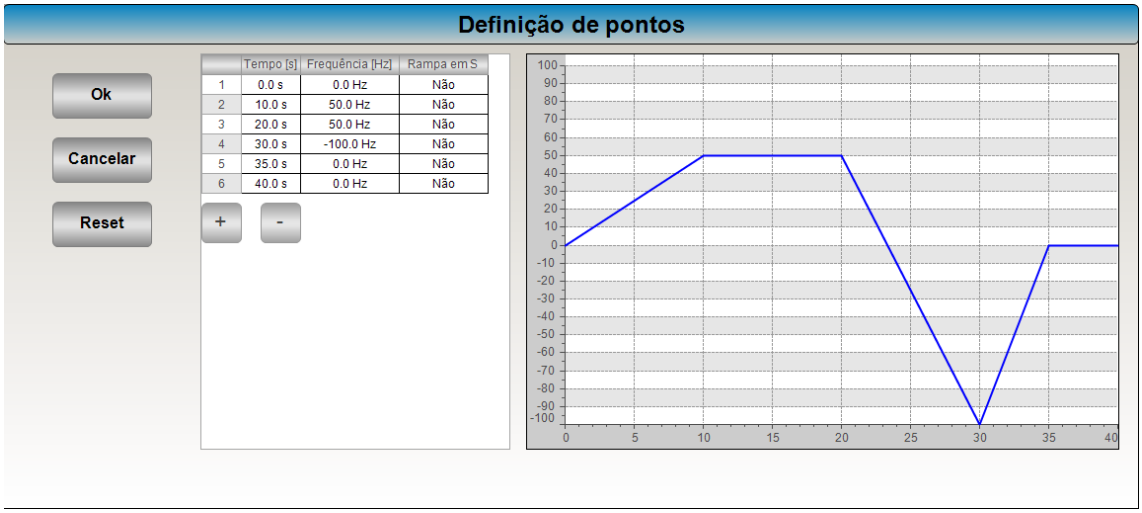

Figura 47 - Criação de um perfil de velocidade com gráfico auxiliar.

Quando uma peça chega ao centro de maquinagem este, sabendo de que peça se trata e em qual deve ser transformada, copia a matriz do perfil de velocidade respetivo para uma outra matriz, idêntica em dimensões, da qual serão lidas as informações a enviar aos variadores. Quando a porta do centro de maquinagem se fecha, o *soft-PLC* começa a transmitir as referências de frequência, por ordem e a seu devido tempo. Também transmite os formatos das rampas e os tempos de aceleração e desaceleração, já devidamente calculados. Após enviar a última referência, que tem sempre como objetivo a paragem do motor, o *soft-PLC* espera que a variável do estado do variador tome o valor 0 para concluir a maquinagem, abrir a porta do centro de maquinagem e proceder com o processo de separação.

5.4. Controlo dos motores passo-a-passo

De forma análoga ao subcapítulo anterior, é importante referir que informação é enviada e recebida pelo *soft-PLC* durante o controlo dos motores passo-a-passo e como é que essa troca de informação se processa. Neste caso foi utilizado um PLC para fazer a ligação entre o *soft-PLC* e os motores e, como tal, torna-se relevante referir as capacidades de controlo de movimento de motores passo-a-passo por parte do PLC e as variáveis a trocar por MODBUS TCP, já que estas foram escolhidas pelo autor, ao contrário do caso dos variadores de frequência, em que são definidas pelo fabricante.

5.4.1. Capacidades de controlo de movimento de motores passo-a-passo

O ambiente de desenvolvimento integrado, TIA Portal, no qual se desenvolve e testa o código de controlo para o PLC Siemens utilizado, permite a definição de objetos tecnológicos. Estes objetos tecnológicos, no contexto de controlo de movimento, nada mais são do que a integração num só conceito de vários parâmetros referentes a um eixo, tais como: tipo de *drive* do motor, relação entre movimento do motor e da carga, limites de velocidade e de deslocamento, etc… O objetivo desta função é simplificar o controlo de movimento da carga, permitindo dar referências ao PLC de valores reias, p. ex. deslocamento da carga em mm, em vez de referências em tensão ou frequência.

No presente caso foi necessário configurar um objeto tecnológico por cada motor, em que o *drive* é acionado por impulsos, escolher quais as saídas físicas que serão usadas para enviar esses impulsos (saídas do módulo de alta frequência), indicar o número de impulsos por rotação do motor e indicar a relação entre o movimento da carga e do eixo do motor (neste caso a relação é de 360º/rotação, porque não há qualquer elemento redutor). Na Figura 48 vê-se os menus do TIA Portal em que é possível configurar os vários parâmetros de um objeto tecnológico de movimento.

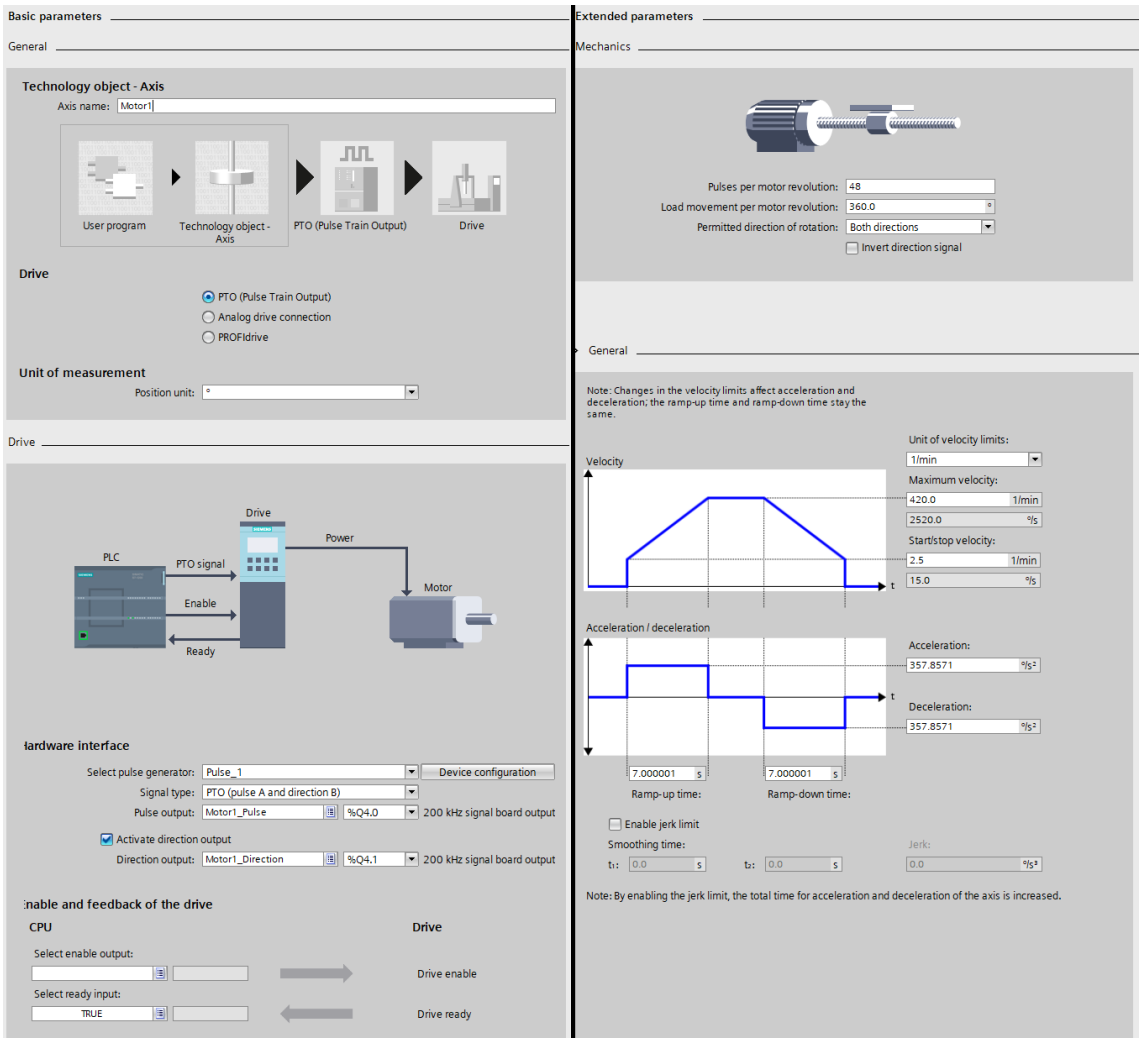

Figura 48 - Configuração de um objeto tecnológico de movimento no TIA Portal.

Configurando os objetos tecnológicos, tornou-se possível o seu controlo através de funções especiais, definidas de acordo com a norma da PLCopen [13], incluídas no TIA Portal. De todas as disponíveis foram usadas apenas três: MC_Power; MC_ChangeDynamic e MC_MoveVelocity. A primeira é obrigatória pois tem como função ativar o objeto tecnológico. A segunda permite mudar os tempos de aceleração e desaceleração, tal como nos variadores de frequência, sendo que a terceira serve para dar uma referência de velocidade e de sentido ao objeto tecnológico, também de forma análoga aos variadores de frequência.

5.4.2. Variáveis de entrada no PLC S7-1200

Tendo em conta as funções de controlo de movimento expostas anteriormente, observa-se que o controlo dos motores passo-a-passo se efetua de forma muito semelhante ao controlo dos motores trifásicos e, assim sendo, as variáveis que o PLC recebe do *soft-PLC* são, para cada motor, as seguintes:

- Ativação do motor (*enable*);
- Sentido de rotação;
- Velocidade de referência;
- Tempo de aceleração;
- Tempo de desaceleração;
- Ativação de meio passo.

Como a comunicação foi, também neste caso, feita recorrendo a um protocolo MODBUS, também se põe a questão do mapeamento das variáveis. No PLC utilizado, o único tipo de memória que é deixado à disposição do utilizador para este configurar são os *holding registers*. Isto não representou um obstáculo, até porque no caso dos variadores de frequência a situação foi igual, já que estes também só suportavam *holding registers*.

Assim sendo, e tal como acontece no caso dos variadores, as variáveis binárias foram todas guardadas no mesmo registo, e as variáveis inteiras usaram o registo na sua totalidade. Essas variáveis binárias são a ativação do motor, o sentido de rotação e a ativação de meio passo, e foram guardadas no mesmo registo nos *bits* 0, 1 e 2, respetivamente.

5.4.3. Variáveis de saída no PLC

As variáveis que o autor decidiu enviar do PLC para o *soft-PLC* são praticamente as mesmas que são usadas no caso dos variadores de frequência:

- Estado do motor;
- Velocidade atual.

A razão para o uso das variáveis prende-se com os mesmos factos apresentados para o caso dos variadores de frequência.

5.4.4. Processamento da troca de informação

O percurso da informação desde o centro de maquinagem virtual até ao PLC é maioritariamente igual ao descrito para o caso dos motores trifásicos. A maior diferença reside no facto de que, após chegarem ao PLC, as informações de tempo e velocidade são

processadas pelas funções especiais para controlo de movimento descritas anteriormente, e estas funções geram impulsos elétricos para os *drives*, nas saídas do PLC especificadas, à frequência correta de forma a que os motores tenham o comportamento dinâmico pretendido. Os *drives*, por sua vez, energizam as bobines corretamente de forma a que os motores deem um passo por cada impulso.

Além disso, as matrizes são diferentes na 3ª coluna, onde em vez de haver informação sobre o formato das rampas de aceleração e desaceleração, tem-se informação sobre a utilização de meio passo.

5.5. HMI

A introdução de uma HMI teve interesse, em primeiro lugar, na medida em que permitiu ao autor o desenvolvimento das suas capacidades no que toca à criação deste tipo de interfaces, sendo que foram exploradas algumas das capacidades e limitações do software utilizado, e em segundo lugar porque permite que o sistema seja mais flexível.

A sua utilização permite que se modifiquem parâmetros do sistema, desde os mais simples como, por exemplo, número de peças a fabricar, até outros mais complexos, como o perfil de velocidade de um motor, que tem de obedecer a determinadas regras.

Por fim, também possibilitou a visualização de dados em tempo real sobre, neste caso, velocidades dos motores e número de peças produzidas.

5.5.1. Disposição dos objetos na HMI

Na Figura 49 está apresentado o menu principal da HMI. Neste é possível, em primeiro lugar, escolher a qual dos centros de maquinagem se quer aceder, utilizando os dois botões no topo. Para cada centro de maquinagem pode-se ativar ou desativar tanto o seu funcionamento como a alimentação de peças.

Há um pequeno menu de definições, a partir do qual se acede às definições sobre tipo de alimentação, número de cada peça a produzir e perfil de velocidade para cada motor. Numa tentativa de o tornar mais agradável ao olhar, este menu foi construído em cascata. No entanto, após a sua criação, concluiu-se que essa decisão não foi a mais acertada porque, apesar de o objetivo ter sido atingido, o processo de criação é muito demorado e propício a erros e, assim sendo, neste caso a estética não compensa o esforço.

Na figura vê-se também uma janela com indicações básicas, mas úteis, sobre o número de peças já produzidas e o número de cada peça a produzir quando o sistema está em modo de alimentação em série.

No fundo da imagem há um espaço destinado à apresentação da velocidade atual dos motores do centro de maquinagem em questão.

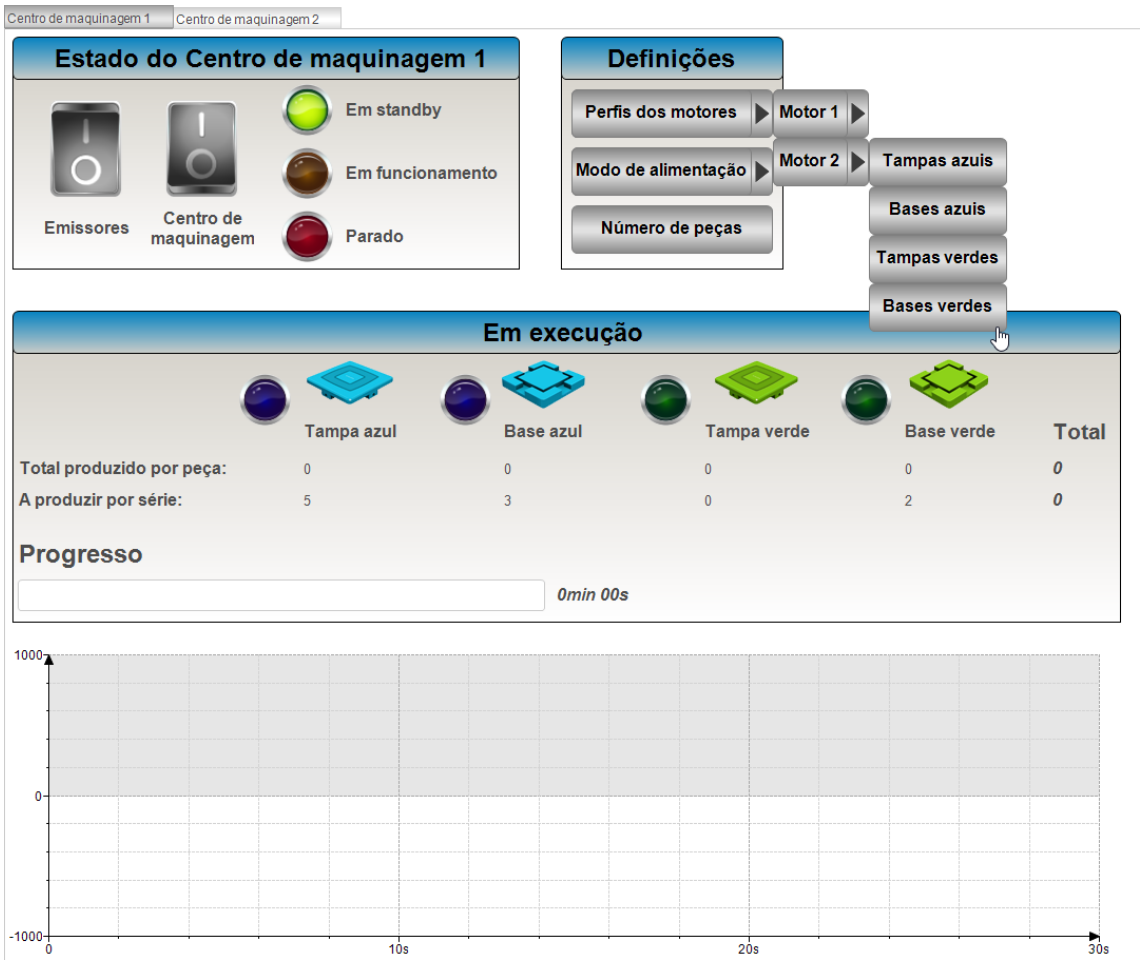

Figura 49 - Menu principal da HMI.

5.5.2. Capacidades de configuração do sistema

A HMI criada tem a capacidade de ligar e desligar alguns componentes, assim como modificar certos parâmetros numéricos. A lista de funções acessíveis pela HMI é a seguinte:

- Ligar/desligar emissores de peças;
- Ligar/desligar centros de maquinagem;
- Alterar entre emissão aleatória e emissão em série;
- Modificar número de peças a produzir por série;
- Modificar perfis de velocidade dos motores em função da peça.

As primeiras três funções mencionadas são muito simples, passando apenas pela modificação de uma variável booleana. A modificação de número de peças a produzir também é uma simples mudança de um valor inteiro. Mas modificar os perfis de velocidade é uma tarefa já bem mais complexa porque implica a mudança de valores numa matriz e obriga à verificação do novo perfil tendo em conta as condições que o tornam válido.

Isto envolveu a introdução de código no botão “Ok” que, quando premido, realiza a verificação das condições para o perfil a ser editado. Além desse, outros botões que

aumentam ou diminuem o número de linhas da matriz usadas para definir o perfil, ou que dão *reset* ao perfil também contêm código.

Uma outra adição que facilita imenso a criação de um novo perfil de velocidade é um gráfico, ao lado da tabela de valores do perfil, que se vai adaptando aos dados inseridos na tabela e mostra uma pré-visualização do que será o novo perfil, como já exibido na Figura 47.

5.5.3. Visualização de dados em tempo real

O interesse da HMI não reside apenas no facto de se poderem alterar parâmetros do sistema, mas também na possibilidade de ver dados relevantes sobre este durante o funcionamento. Por esse motivo foram incluídas algumas funcionalidades de observação do estado do centro de maquinagem e dos motores.

Quanto ao centro de maquinagem, é possível saber se ele está em funcionamento, em espera ou desligado. A HMI também devolve informação sobre que peça está a ser produzida, quantas já foram produzidas, quantas são para produzir por série, indica o progresso na maquinagem da peça atual e quanto tempo já passou desde o início da maquinagem dessa peça. Em relação aos motores, existe um gráfico que indica as suas velocidades.

6. Conclusões e trabalhos futuros

A presente dissertação foi dedicada, acima de tudo, ao desenvolvimento e implementação de soluções de controlo para ambientes industriais, passando-se por modelação, programação e comunicação até se obter, no final, um sistema que funcionasse tal como foi idealizado.

Para tal, foi utilizado o GRAFCET como ferramenta de modelação, revelando-se simples e funcional. A sua utilização, em paralelo com a necessidade de programar PLC's, culminou no desenvolvimento de um método simples e rápido de escrever um código em texto estruturado a partir de um grafcet. A necessidade de implementação de algoritmos de tomada de decisão em código para PLC, aliada ao facto de os códigos de controlo serem sempre criados em linguagens previstas na norma IEC 61131-3, contribuiu para uma formação atualizada do autor em programação de PLC's. Para esse efeito também contribuiu a utilização dos ambientes de desenvolvimento integrado mais populares da atualidade, sendo esses o Codesys e o TIA Portal.

No TIA Portal passou-se pela configuração de comunicações segundo um protocolo aberto moderno, o MODBUS TCP, e pela utilização de uma biblioteca de funções para controlo de movimento, definida de acordo com a norma da PLCopen, que foi utilizado no controlo de motores passo-a-passo. Já no Codesys passou-se pela programação de código de controlo em texto estruturado, pela configuração de comunicações em MODBUS TCP, MODBUS RTU e através de um servidor OPC. Houve ainda a possibilidade de desenvolver uma HMI, necessária em qualquer sistema automático flexível, complementando a programação e a comunicação.

A utilização de dois tipos de motores diferentes foi uma mais valia já que o controlo destes, de um ponto de vista prático, nunca tinha sido abordado ao longo do curso. O controlo dos motores de indução trifásicos foi alcançado pelo uso de variadores de frequência com capacidades de comunicação, enquanto que o dos motores passo-a-passo necessitou de *drives* associados a um PLC para ser realizado.

Considera-se que os códigos criados para o controlo dos vários cenários sucederam no cumprimento do seu objetivo, e os algoritmos de tomada de decisão para os casos de ordenação também se revelaram funcionais. O *software* de simulação Factory I/O revelou-se deveras vantajoso, não só por permitir a criação de cenários virtuais adequados aos objetivos pretendidos, mas também por facilitar os testes dos códigos de controlo desenvolvidos. Em retrospetiva, pode-se dizer que a dissertação correu tal como planeado, e os objetivos propostos foram cumpridos, tendo o autor aprendido nas áreas de programação de PLC's, comunicação industrial, controlo de movimento e criação de HMI's.

6.1. Trabalhos futuros

Na área da programação, pode-se dizer que a geração de código para PLC a partir de um grafcet despertou interesse no autor, assim que se sugere que, cajo haja interesse, se faça uma investigação e análise de métodos já desenvolvidos por outros. A criação de um *software* que realizasse a geração de código de forma automática com base num qualquer método selecionado também poderá ser um tema de interesse neste ramo.

Já na área do controlo de movimento, a execução deste a outro tipo de equipamentos seria, sem dúvida, um tema de interesse. Neste trabalho só foram abordados motores de indução trifásicos e motores passo-a-passo, mas existem muitos outros, cujas caraterísticas certamente implicam a utilização de outros meios que não foram abordados nesta dissertação.

Por fim, no ramo das comunicações industriais, julga-se que a utilização de outros tipos de comunicação seria de potencial interesse.

Bibliografia

1. 3S-Smart Software Solutions. *CODESYS Online Help*. [cited 2018]; Available from: https://help.codesys.com/.
2. Siemens, *TIA Portal Information System*.
3. J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. 2005: Addison-Wesley.
4. José Jácome Felgueiras, *Controlo Distribuído e Heterogéneo de Linhas de Produção Industriais*. Tese de Mestrado em Engenharia Mecânica, opção de Automação, FEUP, 2017.
5. *Langage de spécification GRAFCET pour diagrammes fonctionnels en séquence: norme européenne, norme française NF EN 60848, août 2002*. 2002: UTE.
6. Karl Heinz John and Michael Tiegelkamp, *IEC 61131-3: Programming Industrial Automation Systems Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Aids*. 2010: Springer Publishing Company, Incorporated. 376.
7. International Eletrotechnical Commission, *IEC 61131-3*. 2013. p. 464.
8. Hervé Guéguen and Noël Bouteille, *Extensions of Grafcet to structure behavioural specifications.* Control Engineering Practice, 2001. **9**(7): p. 743-756.
9. R. Zurawski, *The Industrial Information Technology Handbook*. 2004: CRC Press.
10. Robert Julius, Max Schürenberg, Frank Schumacher, and Alexander Fay, *Transformation of GRAFCET to PLC code including hierarchical structures.* Control Engineering Practice, 2017. **64**: p. 173-194.
11. Frank Schumacher and Alexander Fay, *Formal representation of GRAFCET to automatically generate control code.* Control Engineering Practice, 2014. **33**: p. 84-93.
12. Omron Yaskawa Motion Control B.V, *VS mini J7 User's Manual*. 2006.
13. PLCopen, *Function blocks for motion control*. 2.0 ed. 2011. 141.

Anexo A: Grafcets derivados dos programas de controlo do cenário de ordenação horizontal

No capítulo 3 foi explicado o método desenvolvido para a conversão entre GRAFCET e código em texto estruturado para PLC. Visto que esse método é, de certa forma, reversível, pode obter-se um grafcet a partir do código. Esse grafcet é de grande utilidade na compreensão do código e, por isso, é uma ótima forma de documentar os programas criados.

A seguir são apresentados os grafcets criados a partir dos programas desenvolvidos para PLC para o caso do cenário de ordenação horizontal.

A.1. Variáveis de entrada e saída no PLC

Tabela 3: Variáveis de entrada e de saída no PLC

	Variável	Tipo	Descrição
Variáveis de entrada	GVL.scale	Real	Massa detetada pela balança
	GVL.sensor1	Bool	Estado do sensor 1
	GVL.sensor2	Bool	Estado do sensor 2
	GVL.sensor3	Bool	Estado do sensor 3
	GVL.sensor4	Bool	Estado do sensor 4
	GVL.sensor5	Bool	Estado do sensor 5
	GVL.sensor6	Bool	Estado do sensor 6
	GVL.sensor7	Bool	Estado do sensor 7
	GVL.sensor8	Bool	Estado do sensor 8
	GVL.sensor9	Bool	Estado do sensor 9
	GVL.sensor10	Bool	Estado do sensor 10
	GVL.pusherbacklim[1..8]	Array of bool	Estado do sensor de recuo para cada *pusher*
Variáveis de saída	GVL.scalebelt	Bool	Atuação do tapete da balança
	GVL.pusher[1..8]	Array of bool	Atuação de cada *pusher*
	GVL.stopblade[1..7]	Array of bool	Atuação de cada lâmina de paragem
	GVL.belt1	Bool	Atuação do tapete 1
	GVL.belt2	Bool	Atuação do tapete 2
	GVL.belt3	Bool	Atuação do tapete 3

A.2. Programa de controlo da balança – G00_scale

O programa G00_scale diz respeito ao controlo da balança. Na etapa 0 o programa encontra-se em *standby*, à espera de que algo seja colocado na balança. Assim que é detetado um objeto, o programa espera 1 segundo para o objeto colocado estabilizar, de forma a poder fazer uma leitura correta. Após esse tempo de espera, a massa do objeto é registada na variável "*weight*" para poder ser lida por outros programas, e o tapete da balança entra em funcionamento se as condições o permitirem. Após 2 segundos de funcionamento do tapete, o programa volta ao estado de *standby* e espera por outro objeto.

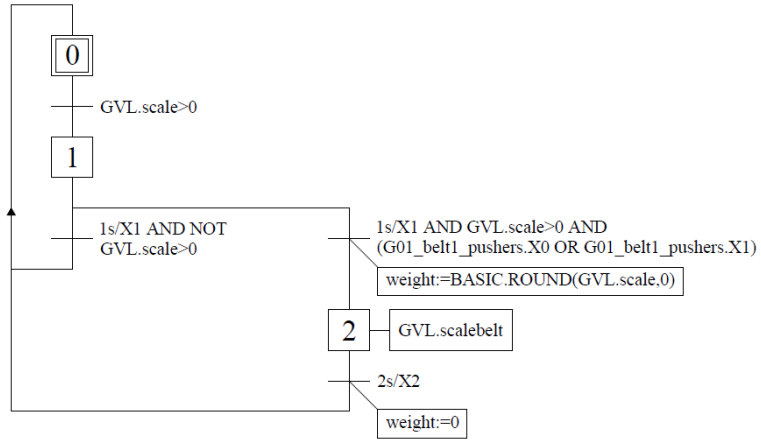

Figura 50 - Grafcet da balança

Tabela 4: Variáveis internas ao programa G00_scale

Variável	Tipo	Descrição
weight	Real	Registo de massa detetada arredondado à unidade

A.3. Programa de controlo do tapete 1 e *pushers* – G01_belt1_pushers

O programa G01_belt1_pushers diz respeito ao controlo do tapete 1 e dos *pushers*. Na etapa 0 o programa encontra-se em *standby*, comutando para a etapa 1, na qual se ativa o tapete 1, quando a balança começa a transportar um objeto. Sempre que a balança envia um novo objeto o programa regista um incremento no número de objetos no tapete 1 e regista também a massa do novo objeto num *array*. Quando todos os objetos estão colocados em frente a um *pusher* o tapete 1 é parado e é consultado o algoritmo de ordenação horizontal que, através do *array* de massas dos objetos, decide quais os *pushers* a ativar. Ativam-se então os devidos *pushers* e modificam-se uma série de variáveis, como o *array* de massas, de forma a que o algoritmo possa ser utilizado corretamente outra vez. Quando todas as peças movidas pelos *pushers* passarem o sensor 9, o algoritmo é consultado outra vez. Este processo repete-se até todos os objetos em frente aos *pushers* terem sido movidos, ou seja, já não sobrar nenhum no tapete 1. Nesse caso, o programa volta à etapa 0. Todas as variáveis internas ao programa podem ser vistas na Tabela 5.

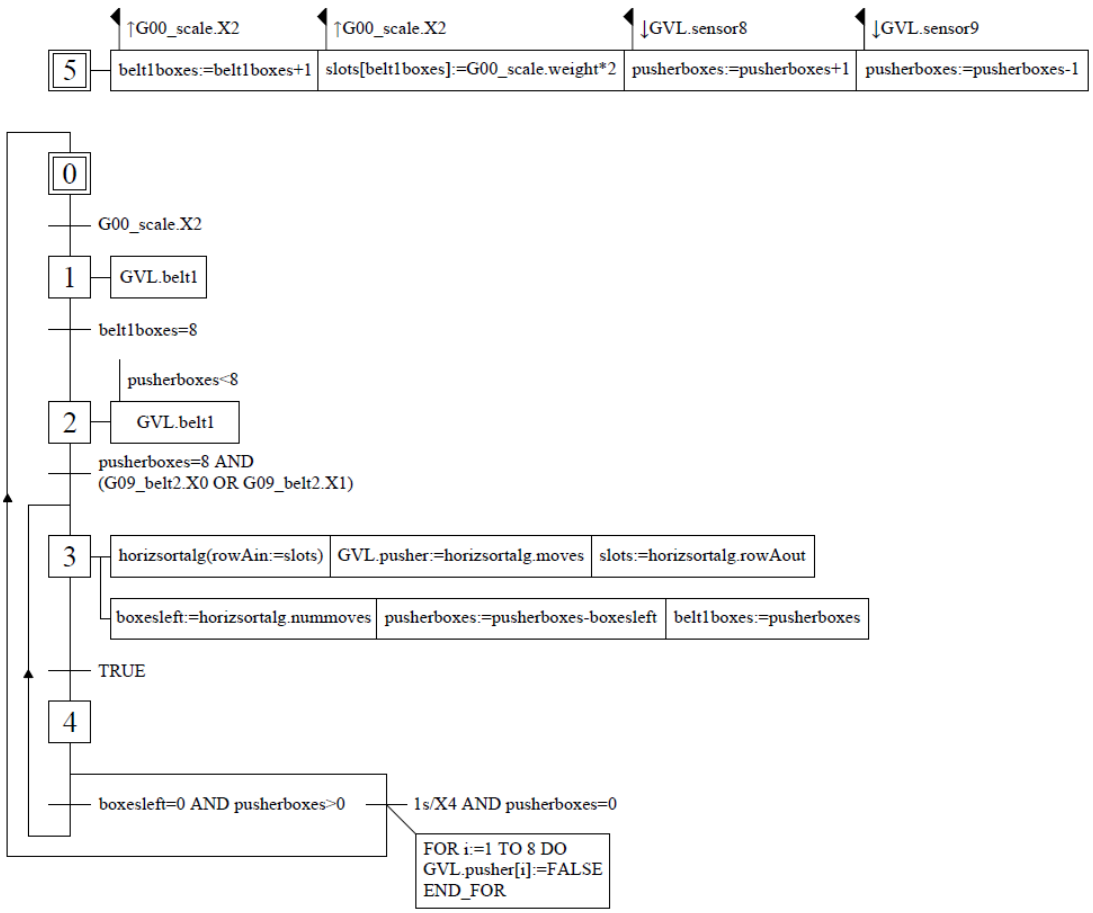

Figura 51 - Grafcet do tapete 1 e pushers

Tabela 5: Variáveis internas ao programa G01_belt1_pushers

Variável	Tipo	Descrição
belt1boxes	Int	Número de caixas no tapete 1
pusherboxes	Int	Número de caixas em frente aos *pushers* no tapete 1
boxesleft	Int	Número de caixas no tapete 2 que ainda não passaram o sensor 9
slots[1..8]	Array of real	Array das massas das caixas em frente aos *pushers* no tapete 1

A.4. Programa de controlo da lâmina de paragem 1 – G02_stopblade1

Todos os programas responsáveis por controlarem lâminas de paragem (G02_stopblade1 até G08_stopblade7) funcionam do mesmo modo. Quando um sensor sofre uma transição descendente, a lâmina de paragem associada a esse sensor é ativada, isto se a lâmina de paragem anterior já estiver ativa. A exceção é a lâmina de paragem 1, pois não há nenhuma anterior a esta. No entanto, para este caso é preciso garantir que o *pusher* 2 esteja recuado, caso contrário este pode atuar o sensor indevidamente no recuo.

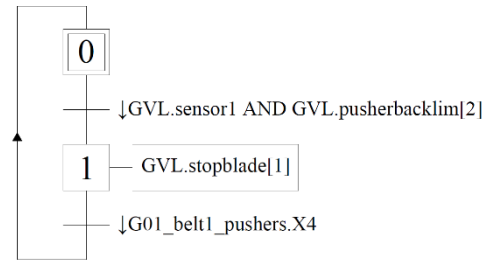

Figura 52 - Grafcet da lâmina de paragem 1

A.5. Programa de controlo da lâmina de paragem 2 – G03_stopblade2

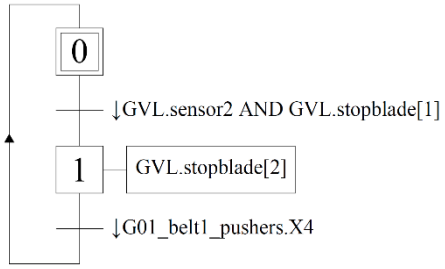

Figura 53 - Grafcet da lâmina de paragem 2

A.6. Programa de controlo da lâmina de paragem 3 – G04_stopblade3

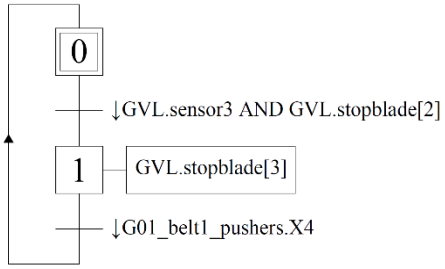

Figura 54 - Grafcet da lâmina de paragem 3

A.7. Programa de controlo da lâmina de paragem 4 – G05_stopblade4

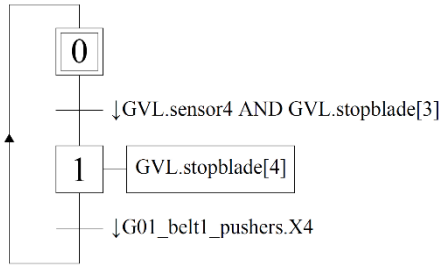

Figura 55 - Grafcet da lâmina de paragem 4

A.8. Programa de controlo da lâmina de paragem 5 – G06_stopblade5

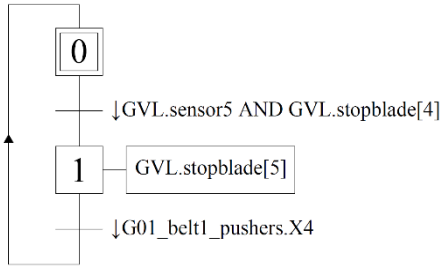

Figura 56 - Grafcet da lâmina de paragem 5

A.9. Programa de controlo da lâmina de paragem 6 – G07_stopblade6

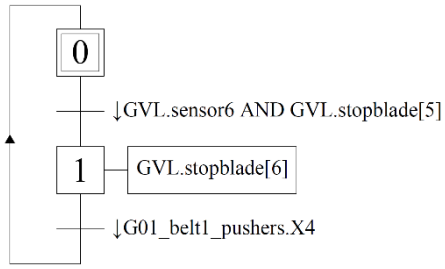

Figura 57 - Grafcet da lâmina de paragem 6

A.10. Programa de controlo da lâmina de paragem 7 – G08_stopblade7

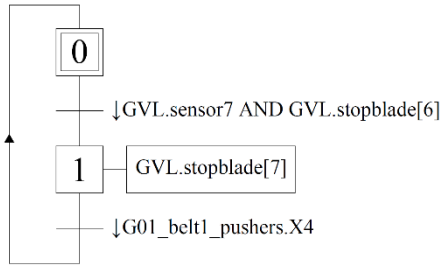

Figura 58 - Grafcet da lâmina de paragem 7

A.11. Programa de controlo do tapete 2 – G09_belt2

O programa G09_belt2 é responsável por controlar o tapete 2. Na etapa 0 o programa está em *standby* e, assim que aparecem objetos à esquerda do sensor 9 (empurrados pelos *pushers*), o tapete 2 ativa-se. Este mantém-se ativado até não haver mais objetos no tapete 2 ou no caso do tapete 3 estar a transportar uma fila compacta de peças para eliminação. Quando não houver mais objetos no tapete 2, o programa volta à etapa 0. Todas as variáveis internas ao programa estão exibidas na Tabela 6.

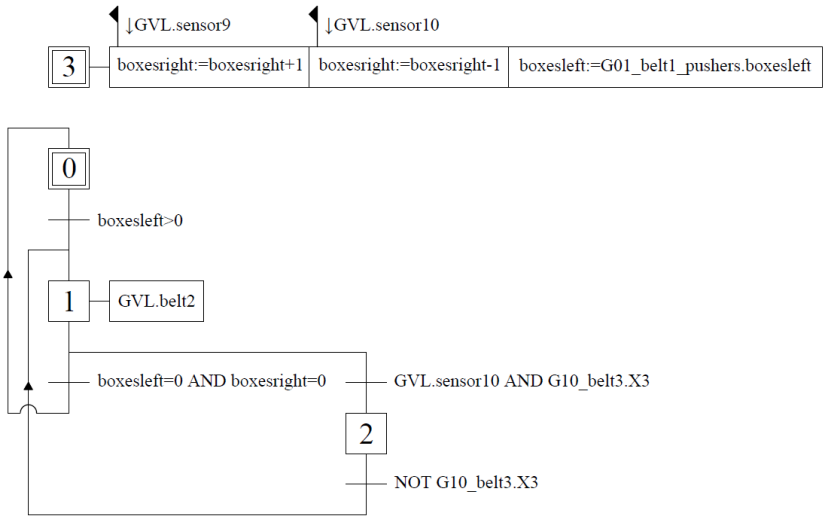

Figura 59 - Grafcet do tapete 2

Tabela 6: Variáveis internas ao programa G09_belt2

Variável	Tipo	Descrição
boxesleft	Int	Número de caixas no tapete 2 antes do sensor 9
boxesright	Int	Número de caixas no tapete 2 após o sensor 9

A.12. Programa de controlo do tapete 3 – G10_belt3

O programa G10_belt3 é responsável pelo controlo do tapete 3. Sempre que um objeto chega ao tapete 3, ou seja, quando o sensor 10 se ativa, o tapete 3 é ativado. Logo que o sensor 10 se desativa, o tapete 3 para. Isto faz com que os objetos se acumulem compactamente em cima do tapete. Quando 8 objetos estão reunidos o tapete 3 avança durante 5s de forma a enviar a fila compacta de objetos para eliminação.

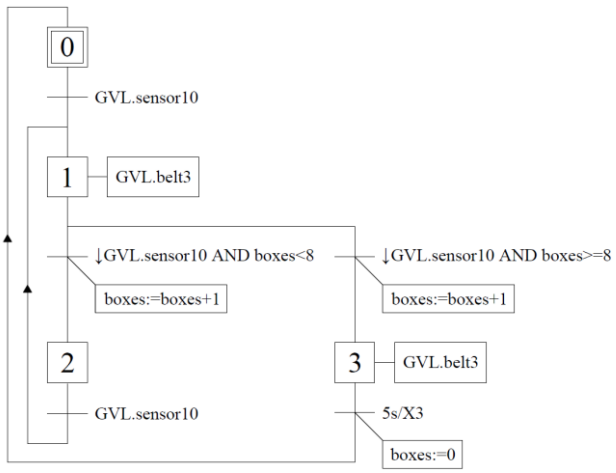

Figura 60 - Grafcet do tapete 3

Tabela 7: Variáveis internas ao programa G10_belt3

Variável	Tipo	Descrição
boxes	Int	Número de caixas no tapete 3